Hans Werner

Mikrocomputer

Eine kleine Einführung

7. Teil

Zusammenfassung: Hardware

In kurzer Form soll der Aufbau des Mikrocomputers wiederholt werden. Rechenwerk, Akkumulator und alle Register (außer ER und AR) werden zu einem großen Funktionsblock – dem Mikroprozessor – zusammengefaßt.

Der Prozessor wird auch als CPU (Central Processing Unit) bezeichnet. Eine einheitliche Festlegung, welche Funktionsgruppen in der CPU sind, gibt es nicht. Mikroprozessor und periphere Einheiten, wie Speicher usw., bilden zusammen den Mikrocomputer.

4 Busse ermöglichen Datentransport und Steuerung:

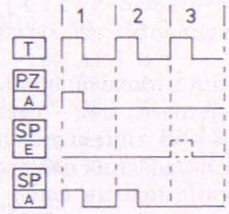
1. Eingabe-Bus,
durch den Daten aus dem Speicher oder aus dem Eingangsregister in die CPU gelangen.

2. Ausgabe-Bus,
durch den Daten aus der CPU in den Speicher oder in das Ausgangsregister können.

3. Adreß-Bus,
zum Adressieren von Speicher und Ein-Ausgangsregister.

4. Steuer-Bus,
zum Steuern der peripheren Einheiten.

Was meldet der Steuerbus?



Takt:
Intervalle 1...3

PZ, A:
Adresse an Speicher,
Auslesen des Befehls.

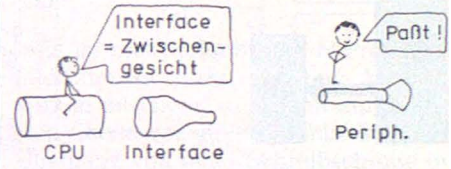
SP, E:
Eingabe an Speicher bzw. Ausgaberegister.
Kommt nur bei Befehl SP.

SP, A:
1. Impuls: wie bei PZ, A

2. Impuls: Inhalt der adressierten Speicherzelle (bzw. ER) wird in den Akku geladen.

Die CPU läßt sich nicht immer direkt an andere Bausteine anschließen, z. B. ist für die Register ER und AR eine zusätzliche Logikschaltung notwendig.

Die Schnittstelle zwischen zwei Funktionsgruppen wird als Interface bezeichnet, die Anpassungsmaßnahmen als Interface-Logik.



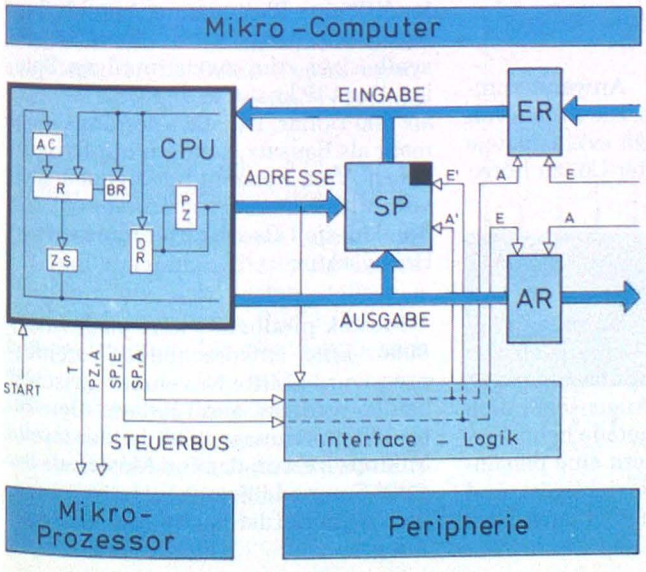
Zusammenfassung: Software

Mit Software ist in diesem Fall nur der Befehlsvorrat gemeint. Er beschreibt die Hardware-Eigenschaften des gesamten Systems.
Eingeklammerte Ausdrücke bezeichnen wiederum den Dateninhalt des Bausteines.

AD n SU n	000 001	Addieren Subtrahieren	(AC) + (n) (AC) − (n)	Ergebnis steht nach	1)
UN n OD n	010 011	log. UND log. ODER	(AC) ∧ (n) (AC) ∨ (n)	Ausführung im AC	
LA n SP n	100 101	Laden Speichern	(n) in AC (AC) in n		2)
JP n ST	110 111	Sprung Stop	(n+1) in PZ	wenn (AC)≠0	2)3)
LA31 SP31	100 101	Eingabe Ausgabe	(ER) in AC (AC) in AR	nur mit Adresse 31	2)

Bemerkungen:
1) n steht für Speicherzelle (n muß ungleich 31 sein)
2) Akkuinhalt geht nicht verloren
3) Springt immer einen Schritt weiter als angegeben

Mit dem kleinen Befehlsvorrat des Computer-Modelles sind durchaus nette, sinnvolle Programme möglich. Natürlich werden leicht überschaubare Beispiele gewählt. Für solche Probleme lohnt sich der Einsatz eines Mikrocomputers nicht, obwohl sie als Teilproblem in größeren Programmen enthal-

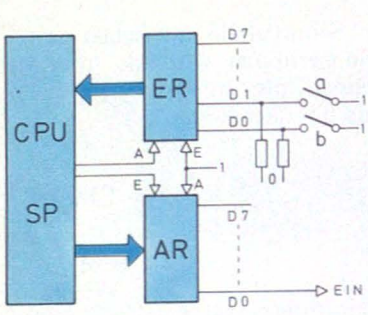


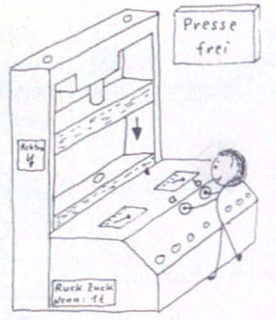
ten sein können. Um der Langeweile vorzubeugen, wird zwischendurch die Hardware vereinfacht. Der Befehlsvorrat bleibt erhalten.

Handbedienung

Eine Presse wird von Hand bedient. Um den Bediener zu schützen, darf sie sich nur durch ständiges Drücken von zwei Tastern a und b einschalten lassen.

Die „Reaktionsgeschwindigkeit" des Mikrocomputers soll sehr viel größer als die des Bedieners sein, d. h. die Länge des Programms spielt eine untergeordnete Rolle.





Beide Schalter liegen am Eingangsregister. Der Ausgang D0 des Ausgaberegisters ist zur Presse geführt. ER,E und AR,A sind fest auf 1 geschaltet. Damit gelangen die Schalterstellungen ständig in das ER und das Ergebnis ständig aus dem AR zur Presse.

Beim 1. Durchlauf des Programms wird ein falscher Wert in Kauf genommen. Das Problem kann z. B. durch einen Nullstelleingang am Ausgangsregister gelöst werden.

Das eigentliche Programm beginnt mit dem Ladebefehl LA 31, der alle 8 bit des ER in den Akku transportiert und nicht die einzelnen Schalterstellungen a, b. Sie können durch UND-Verknüpfung mit der binären 1 bzw. 2 eliminiert werden.

Schalterstellung a

```
                ba    X,Y kann 0
Eingabe: 00000XY    oder 1 sein.
   1   : 00000001   Nur Y bleibt
UND    : 0000000Y   erhalten.
```

Die 1 soll schon in Zelle 28 gespeichert sein, die 2 wird durch Addition gewonnen.

```
 0   LA 28    1 in Akku laden
 1   AD 28    1 + 1
 2   SP 27    2 in Zelle 27 speichern
 3   LA 31    Eingabe
 4   SP 30    Schalterstellungen nach Zelle 30
 5   UN 28    Schalterstellung a eliminieren
 6   SP 29    a in Zelle 29 speichern
 7   LA 30    Schalterstellungen aus 30 in AC
 8   UN 27    Schalterstellung b eliminieren
 9   UN 29    b UND a
10   SP 31    Ausgabe     Ergebnis nach AR
11   LA 28    damit AC-Inhalt ungleich 0 ist
12   JP  2    Schalterabfrage wiederholen
```

Mit Befehl 11 wird nur vermieden, daß der Akkuinhalt 0 und der Sprungbefehl unwirksam ist. Unser Modell hat keinen unbedingten Sprungbefehl, d. h. er wird nicht unabhängig von einer Bedingung (z. B. Akkuinhalt) ausgeführt.

Erzeugen von 0...

...durch Subtraktion gleicher Zahlen. Der Trick ist bei richtigen Mikrocomputern unnötig.

```
LA n
SU n
```

(Fortsetzung folgt)

µC-Nachrichten

Die Firma Integrated Computer Systems (**ICS**) hat in München, Leonrodstraße 58, eine Vertretung eröffnet. Zweck ist vor allem die Durchführung von Kursen und die Unterstützung beim Selbststudium nach dem Motto „Education is our business".

□

Der Mikrocomputer **Apple II** wird unter Lizenz bald auch von *ITT* in Europa gefertigt. Ein entsprechendes Abkommen, in dem es um einige Millionen Dollar geht, wurde kürzlich zwischen ITT und Apple unterzeichnet. Es betrifft auch den Verkauf einer nicht begrenzten Stückzahl durch ITT, allerdings nur in Europa. Beide Firmen haben auch vereinbart, Anwenderprogramme auszutauschen. Die Rechte von ITT sind allerdings nicht exklusiv, wie von der Apple Computer Co. zu hören war.

□

Radio Shack alias Tandy plant, in den USA in den nächsten Monaten mindestens etwa 50 „Computer Centers" zu eröffnen. Ein Vorversuch war offensichtlich der Computer-Shop von Tandy in Ft. Worth. Augenscheinlich hat man erkannt, daß gerade beim Verkauf von Heimcomputern eine persönliche Beratung überaus wichtig ist, und dazu braucht man ein dichteres Netz von Läden.

Auf der Personal Computing Show in Philadelphia (24.–27. August) stellte *RCA* als Zubehör zu dem **1802-Computer VIP** eine Platine zum Anschluß an Farbfernsehgeräte und einen Musiksynthesizer zum zweistimmigen Spielen. Der VIP kostet in den USA weniger als 300 Dollar, ist jetzt allerdings nicht mehr als Bausatz, sondern nur fertig zu haben. Vom Konzept her eignet er sich vor allem zum Programmieren von Spielen, zielt also in erster Linie auf den Hobbysektor.

□

Mostek produziert jetzt auch Intel's **8086.** Eine entsprechende Vereinbarung wurde Mitte November zwischen beiden Firmen geschlossen. Der 16-bit-Mikroprozessor 8086 ist der zweite Mikroprozessor, der von Mostek als Second-Source-Lieferant gefertigt wird; im Programm ist bereits der Z 80 von Zilog.

Eckart Märkel

EPROM-Programmiergerät

1. Teil

Nichtflüchtige Speicher, schon immer ein entscheidender Bestandteil digitaler Schaltungstechnik, besonders der Rechnertechnik, haben in Form von ROM, PROM und EPROM besonders mit dem Aufkommen der Mikrocomputer erheblich an Bedeutung gewonnen. Für den Hobbyisten sind besonders PROM und EPROM interessant, da diese in Einzelstücken mit einem individuellen Bitmuster geladen werden können. Das EPROM hat darüber hinaus noch die Eigenschaft, mit UV-Licht löschbar zu sein.

Ein großes Hindernis für die Anwendung von EPROMs im Hobby-Bereich ist jedoch die relativ komplizierte Programmierung von (E)PROMs verglichen mit dem Schreiben in ein RAM. Zwar bieten die Hersteller alle passende Programmiergeräte für ihre Produkte an, aber leider nur zu sehr hohen Preisen. Teuer und umständlich ist auch der Weg, die Programmierung in Auftrag zu geben, wenn das nicht schon am Erstellen des Datenträgers scheitert.

Diesem Übel soll die nachfolgende Bauanleitung für ein EPROM-Programmiergerät abhelfen.

Für die Hobby-Anwendung wird es wohl zunächst einmal ausreichen, sich auf eine Typenfamilie zu beschränken und ein Weniger an Bedienungskomfort in Kauf zu nehmen, da das Gerät in der Regel nur gelegentlich benutzt werden wird.

Das vorliegende Programmiergerät ist für die Programmierung der bekannten EPROM-Typen 2704 und 2708 (identisch mit 8708 bzw. SAB 8708) ausgelegt [1, 2]. Beide Typen unterscheiden sich nur in der Speicherkapazität:

2704: Organisation 512 x 8 bit
2708: Organisation 1024 x 8 bit

Bild 1 zeigt das Anschlußbild und das Logikdiagramm dieses Speichers, *Tabelle 1* die wichtigsten statischen Kenn- und Grenzdaten. *Bild 2* stellt das Impulsdiagramm für den Lesebetrieb dar.

Das Schreiben gestaltet sich etwas komplizierter. Bei unprogrammiertem bzw. gelöschtem Baustein sind alle Bits auf „1". Mit Hilfe des Eingangs CS/WE kann eine dieser drei Betriebsarten ausgewählt werden:

$\overline{CS}/WE = V_{ss}$ (0 V) : Lesen
$\overline{CS}/WE = V_{cc}$ (+5 V) : inaktiv, Ausgänge hochohmig
$\overline{CS}/WE = V_{DD}$ (+12 V): Schreiben, Ausgänge wirken als Eingänge

Dieses Programmiergerät benutzt nur die Zustände „Lesen" und „Schreiben". Das Impulsdiagramm *Bild 3* zeigt, daß zum Schreiben zunächst die Schreibfreigabe zu geben ist, dann sind die Adressen 0...1023 nacheinander durchzuzählen, wobei während jeder Adresse auf den Programmiereingang ein 26-V-Programmierimpuls mit definierter Dauer und definierter Anstiegs- und Abfallzeit zu geben ist. Eine solche Programmierschleife ist N-mal zu wiederholen, d. h. so oft, bis

$$N \times t_{pw} \geqq 100 \text{ ms}$$

erfüllt ist (t_{pw} = Programmierimpulsdauer).

Es ist nicht erlaubt, die N Programmierimpulse zuerst auf eine Adresse wirken zu lassen und dann zur nächsten überzugehen! Weiterhin ist beim Übergang von einer Schreibschleife in den Lesezustand das Schreibsignal (WE) noch während des Anliegens der zuletzt programmierten Adresse zurückzunehmen.

Das Programmieren

Bild 4 zeigt nun eine geeignete Schaltung, die die Signale zum Programmieren der EPROMs 2704 und 2708 liefert. Die Stromversorgung ist mit Ausnahme des für den Programmierimpulsgenerators notwendigen Teils nicht mit eingezeichnet. Die Spannungen −5 V und +12 V werden ausschließlich für das EPROM benötigt. Von der +5-V-Stromversorgung werden etwa 600 mA aufgenommen. Eventuell kann ein Teil der Betriebsspannungen dem Gerät

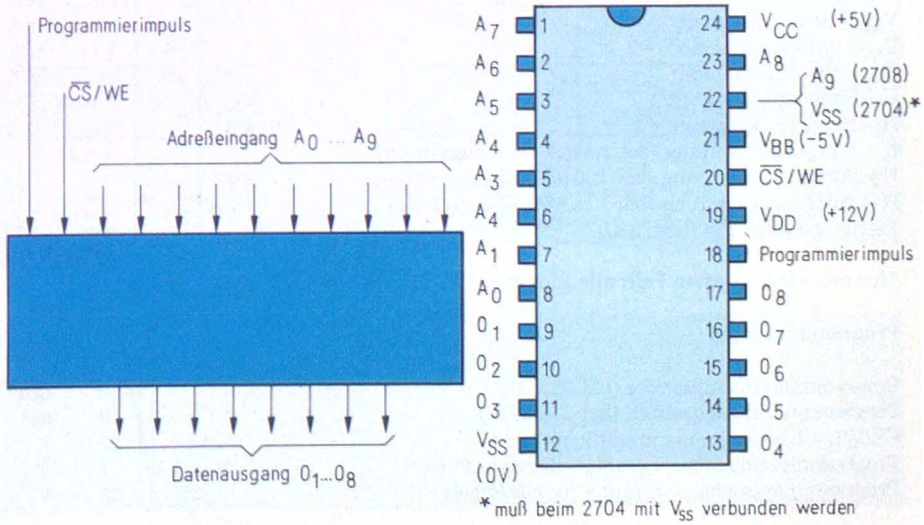


Bild 1. Anschlußbild und Logikdiagramm der Speicher 2704 und 2708

entnommen werden, in dem das EPROM später eingesetzt wird.

An die entsprechend bezeichneten Daten- und Adreßanschlüsse ist eine (statische) Speicherschaltung (RAM) anzuschließen, in die bereits die zu programmierenden Daten eingegeben sein müssen. Natürlich kann es sich dabei auch um ein ROM oder ein anderes PROM handeln.

Gemäß Herstellerangaben sollen alle „don't-care"-Bits auf „1" gesetzt werden. Die Freigabeeingänge des externen Speichers müssen auf „Freigabe" geschaltet sein, der Schreibeingang auf „Lesen", so daß am Dateneingang des Programmiergerätes jederzeit der durch die Adreßausgänge des Programmiergerätes bestimmte Speicherinhalt anliegt.

Bevor jetzt auf die Einzelheiten der Schaltung näher eingegangen wird, soll vorher die Bedienung geschildert werden.

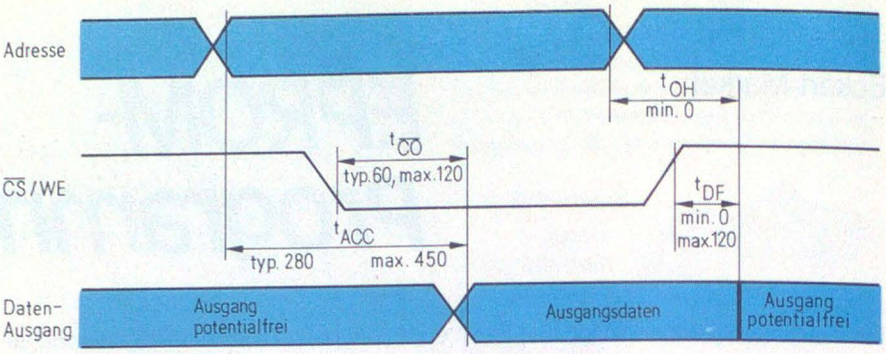


Bild 2. Impulsdiagramm für den Lesebetrieb des 2708 (2704); alle Zeitangaben in ns

Nachdem man den externen Speicher angeschlossen hat, nimmt man das Gerät in Betrieb:

1. EPROM einstecken, Rückstellschalter in Stellung „Rückstellen" bringen, Typenwähler auf „2704" bzw. „2708" einstellen, Betriebsart „Lesen" einstellen;
2. Betriebsspannungen −5 V, +5 V, +12 V einschalten (Achtung: Würde +12 V vor +5 V eingeschaltet, während die Betriebsart „Schreiben" eingestellt ist, würde vorzeitig ein Schreibfreigabe-Signal auf das EPROM gelangen).

Adreß- und Programmierschleifenzähler sowie „Lauf" stehen jetzt auf Null (zugehörige Leuchtdioden aus), das Gerät ist im Ruhezustand. Die Datenleuchtdioden zeigen den Inhalt des Speicherwortes mit der Adresse 0 des EPROM an, leuchten also alle für „11111111" bei unprogrammiertem EPROM.

Zum Programmieren fährt man wie folgt fort:

3. Taktgeber in Stellung „ein" bringen, falls noch nicht geschehen, Schleifenzahl N einstellen (N = 513 normal), Betriebsart „Schreiben" einstellen;
4. Programmierspannung einschalten;
5. Rückstellung in Stellung „nicht rückstellen" bringen;
6. „Start" drücken.

Die Leuchtdiode „Lauf" geht nun an, und an den anderen Leuchtdioden kann man das Durchzählen der Adressen und der Programmierschleifen erkennen. Wegen der Geschwindigkeit ist an den Datenleuchtdioden nur ein Flackern zu erkennen. Während des Programmierens darf keinesfalls der Takt angehalten werden, etwa durch Abstellen des Taktgenerators oder Drücken von „Handtakt", weil man damit einen Programmierimpuls auf eine unzulässige Dauer verlängern könnte. Auch das Rückstellen sollte vermieden werden, weil dann nicht die im Impulsdia-

Tabelle 1: Statische Kenn- und Grenzdaten der EPROMs 2704 und 2708

Grenzdaten			
Betriebstemperatur		−25...+85 °C	
Alle Eingangs- und Ausgangsspannungen (außer Programmiereingang)	bezogen auf V_{BB}	+15...−0,3 V	
Programmiereingangsspannung	bezogen auf V_{BB}	+35...−0,3 V	
Versorgungsspannungen V_{CC} und V_{SS}	bezogen auf V_{BB}	+15...−0,3 V	
Versorgungsspannung V_{DD}	bezogen auf V_{BB}	+20...−0,3 V	
Leistungsverbrauch		+1,5 W	
Betriebsbedingungen			
Temperatur		0...+70 °C	
V_{SS}		0 V	
V_{BB}		−5 V ±5 %	
V_{CC}		+5 V ±5 %	
V_{DD}		+12 V ±5 %	
Kenndaten	min.	typ.	max.
Adreß- u. Bausteinauswahleingangsstrom (bei Eingangsspannung +5,25 V)		1	10 µA
Ausgangsleckstrom (bei +5,25 V Ausgangsspannung und $\overline{CS}$/WE = +5 V)		1	10 µA
V_{DD} – Stromaufnahme*		50	65 mA
V_{CC} – Stromaufnahme*		6	10 mA
V_{BB} – Stromaufnahme*		30	45 mA
L – Eingangsspannung	V_{SS}		+0,65 V
H – Eingangsspannung	+3		V_{CC}+1 V
L – Ausgangsspannung (bei 1,6 mA Ausgangsstrom)			+0,45 V
H – Ausgangsspannung (bei 100 µA Ausgangsstrom)	+3,7		V
H – Ausgangsspannung (bei 1 mA Ausgangsstrom)	+2,4		V
Leistungsverbrauch (bei 70 °C)			800 mW

*für den schlechtesten Fall: alle Eingänge „H", $\overline{CS}$/WE = +5 V, 0 °C

Programmierdaten			
	typ.	max.	
Programmiereingangsstrom (bei 0...1 V)		−3	mA
Programmiereingangsstrom (bei 25...27 V)		20	mA
$\overline{CS}$/WE – Eingangsspannung (Schreiben)	11,4	12,6	V
Programmiereingangsspannung (niedriger Pegel)	V_{SS}=0 V	1	V
Programmiereingangsspannung (hoher Pegel)	25	27	V

(Die Differenz zwischen hohem und niedrigem Pegel muß dabei mindestens 25 V betragen)

gramm Bild 3 geforderte Art des Schreib-Lese-Übergangs eingehalten würde.

Möchte man den Programmiervorgang vorzeitig beenden, stellt man einfach den Programmierschleifenwähler auf „1"; mit dem Ablauf der gerade laufenden Programmierschleife wird die Programmierphase dann normal beendet.

Ansonsten geht das Programmieren nach einigen Sekunden oder normalerweise Minuten, je nach gewählter Schleifenzahl, zu Ende, wobei die Leuchtdiode „Lauf" erlischt.

War das Programmieren für das erste Byte erfolgreich, so wird es nun richtig an den Datenleuchtdioden angezeigt. Natürlich interessiert auch, ob auch alle anderen Bits richtig programmiert wurden. Dazu kann ein Prüflauf durchgeführt werden. Vorher muß aber wieder auf „Lesen" geschaltet werden:

7. Rückstellung in Stellung „Rückstellen" bringen;
8. auf „Lesen" schalten, Programmierspannung abstellen.

Prüfen des Speicherinhaltes

Zum Prüfen verfährt man so:

1. und 2. Falls nicht ohnehin schon geschehen, Inbetriebnahme wie oben;
3. Takt einschalten, Betriebsart „Lesen" wählen, Schleifenzahl 1 einstellen, Rückstellung abstellen;
4. „Start" drücken.

Die Art des Durchlaufes ist wie beim Programmieren, nur daß hier die Inhalte beider Speicher *verglichen* werden. Dazu reicht natürlich eine Schleife aus. Während des Lesens ist es erlaubt, jederzeit durch Rückstellen abzubrechen oder durch Abschalten des Taktes oder Drücken von „Handtakt" zu unterbrechen. Die Datenleuchtdioden geben den EPROM-Inhalt zur angezeigten Adresse an. Möchte man von Hand takten, so stellt man den Taktgenerator ab. Bei jedem vierten Takt geht die Adresse um 1 weiter. Beim Starten muß „Start" während eines Taktimpulses gedrückt sein. Ein normaler Durchlauf geht wie beim Programmieren zu Ende.

Tritt beim Vergleich aber eine Nichtübereinstimmung auf, hält das Gerät sofort an. Angezeigt werden dann das fehlerhafte Byte und die zugehörige Adresse. (Zur Identifikation des fehlerhaften Bits wäre eventuell eine zweite Leuchtdiodeneinheit am Ausgang des externen Speichers vorteilhaft.) Durch Betätigen der Rückstellung gelangt man wieder in den Ruhezustand und kann, falls nötig, die Programmierung wiederholen. Abgeschaltet wird sinngemäß in umgekehrter Reihenfolge wie beim Einschalten.

Die Schaltung

Im Mittelpunkt der Schaltung steht das EPROM 2708 bzw. 2704. Seine Adreßeingänge sind fest mit dem Adreßzähler verbunden, und dessen Ausgänge stehen über einen Anschluß zugleich dem externen Speicher zur Verfügung. Weiter sind daran die Leuchtdiodentreiber (Inverter A 0 bis A 9) und die Schlußerkennungslogik angeschlossen. Der Typenumschalter 2704/2708 schaltet Pin 22 des EPROM entweder auf Masse oder den Adreßzähler. Der Adreßzähler Z 0 bis Z 2 be-

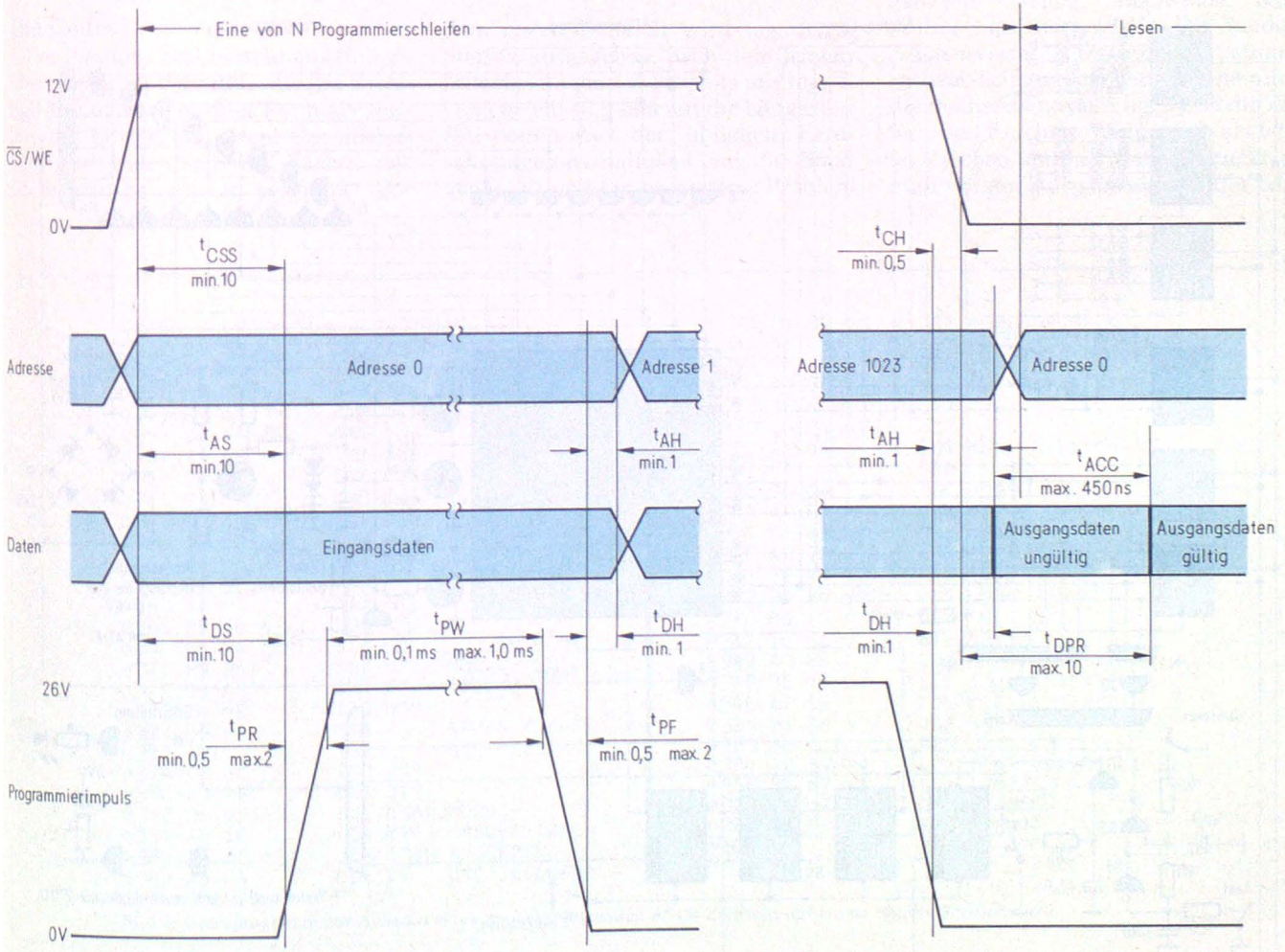


Bild 3. Impulsdiagramm für das Schreiben (Programmieren); alle Zeitangaben in µs, wenn nicht anders vermerkt

steht aus drei Binärzählern 7493; er findet seine direkte Fortsetzung im Schleifenzähler Z 3 bis Z 5, dessen Takt je nach Typenwahl am Adreßzähler passend abgegriffen wird. Auch hier besorgen Leuchtdioden mit den Invertern A 10 bis A 21 eine binäre Anzeige.

Die Datenanschlüsse des EPROMs sind über Tristate-Gatter 74125 (T 1 bis T 8) mit den Dateneingängen des Programmiergerätes verbunden. Während des Programmierens werden diese Tristate-Gatter geöffnet und legen die eingegebenen Daten an die nun zu Eingängen gewordenen EPROM-Anschlüsse. Eine Vergleichslogik meldet mit einem hohen Pegel am Ausgang des NAND-Gatters 7430 (N 4) Ungleichheiten zwischen EPROM-Anschlüssen und Schaltungseingang. Beim Schreiben wird durch die Tristate-Gatter natürlich Gleichheit erzwungen. Die Inverter B 1 bis B 8 werden hier sowohl als Leuchtdiodentreiber als auch in ihrer logischen Funktion betrieben.

In Bild 4 unten ist der Taktgeber eingezeichnet. Er ist so dimensioniert, daß sich für die Programmierimpulsdauer

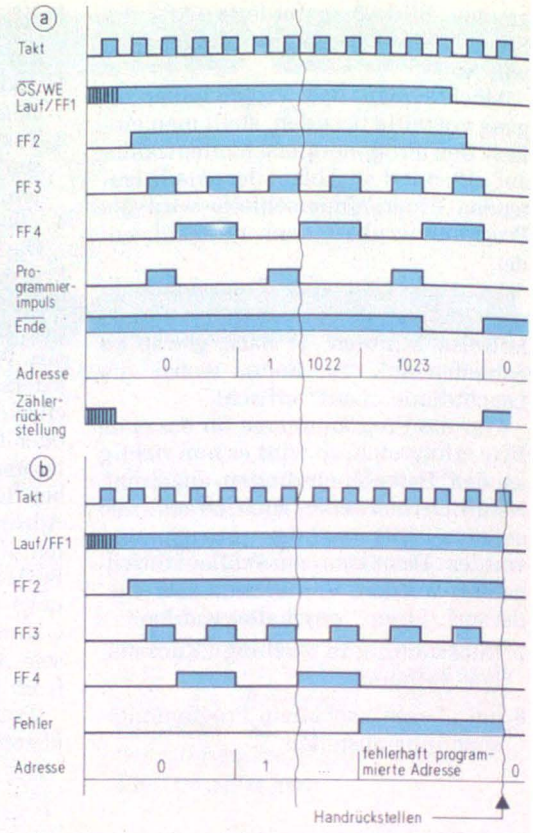
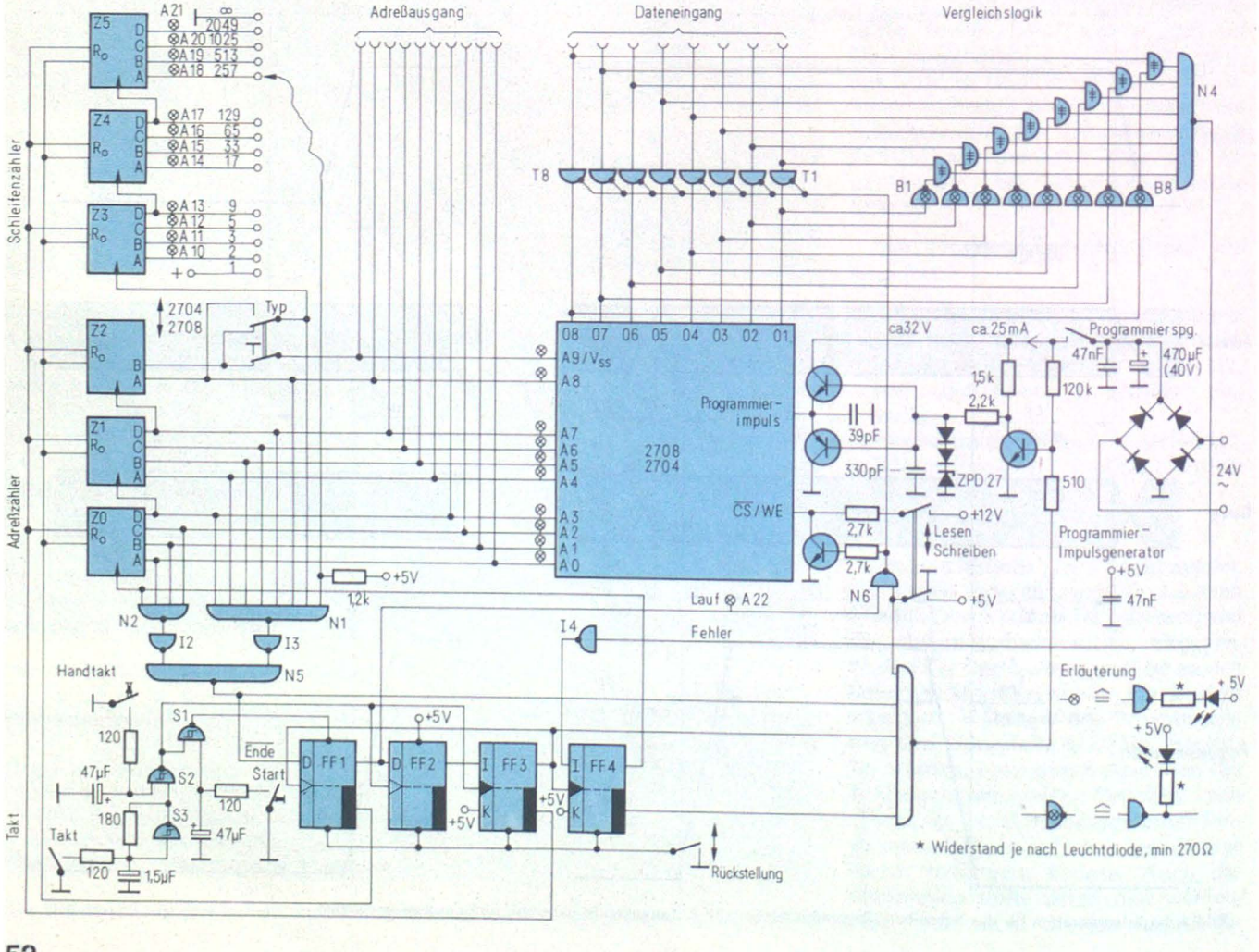


Bild 5. ▶ Impulsdiagramm des Programmiergerätes: (a) für das Programmieren, (b) für das Prüfen

▼ Bild 4. Schaltung des Programmiergerätes. Die EPROM-Adresseneingänge sind fest mit dem Adreßzähler verbunden; ihr Zustand kann binär an Leuchtdioden abgelesen werden



t_{pw} ein mittlerer Wert (0,3 ms) ergibt; dementsprechend sind mindestens 333 Programmierschleifen nötig. Eine genaue Messung kann man ja leicht mit den Zählern unter Zuhilfenahme einer Uhr durchführen (die den Zählern vorgeschalteten Flipflops teilen durch 4). Bei Handtaktung sorgt der Schmitttrigger S 2 zusammen mit dem 47-µF-Elektrolytkondensator für eine Entprellung. Der Widerstand begrenzt nur den Entladestrom. Mit dem Rückstellschalter werden alle Flipflops, indirekt auch die Zähler, zurückgesetzt.

Beim Betrieb ergibt sich nun folgender Ablauf:

Mit der Starttaste wird das D-Flipflop FF 1 gesetzt und die Leuchtdiode „Lauf" an Inverter A 22 leuchtet auf. Falls am Schreib/Lese-Wähler „Schreiben" eingestellt wurde, wird jetzt über das NAND-Gatter N 6 das Signal WE („Write Enable") an das EPROM gegeben, die Tristategatter werden geöffnet. In jedem Falle wird die Rückstellung der Zähler aufgehoben. Mit der nächsten positiven Flanke des Taktimpulses wird das D-Flipflop FF 2 gesetzt, wodurch die Zählschaltung aus den IK-Flipflops FF 3, FF 4 freigegeben wird. Diese erzeugen jetzt das in *Bild 5a* für das Programmieren bzw. *Bild 5b* für das Prüfen dargestellte Impulsdiagramm.

Um eine Programmier- oder Leseschleife zu beenden, muß FF 1 gelöscht werden. Dies bewirkt die Schlußerkennungslogik aus den Gattern N 5, I 2, I 3, N 1 und N 2, die zusammen ein Nandgatter bilden, indem sie Null an den D-Eingang von FF 1 legt. Sie ist nötig, weil das Schreibsignal WE schon zurückgenommen werden muß, während noch die letzte programmierte Adresse anliegt. Damit hängt auch die Stufung des Schleifenwählers zusammen:

$$\text{Schleifenzahl} = 2^m + 1$$

mit $m \in \{0, 1, 2 \ldots 11\}$.

(Schluß folgt)

Baudot-Ausgabeprogramm für den 6502

Die beiden am weitesten verbreiteten Codes zur seriellen Datenübertragung sind der Baudot- und ASCII-Code. Unglücklicherweise arbeiten praktisch alle Mikrocomputer im ASCII- und alle Fernschreiber im Baudot-Code; doch stellt dies keinen Grund dar, ein Programm nicht auch einmal von einem preiswert erhältlichen, gebrauchten Fernschreiber ausdrucken zu lassen, wie der folgende Beitrag zeigt.

Die Codes

Der Baudot-Code besteht aus fünf zu übertragenden Datenbits, die das jeweilige Zeichen in serieller Form als Nullen und Einsen enthalten. Das niederwertigste Datenbit wird zuerst, das höchstwertige zuletzt gesendet. Vor dem ersten Datenbit wird log. 0 als Startbit ausgegeben, nach dem letzten Datenbit folgen 1,5 Stopbits mit log. 1. (1,5 bezieht sich hier auf die Länge; ein Bit dauert bei der üblichen Fernschreibgeschwindigkeit von 50 Baud genau 20 ms.) Ein besonderes Problem des Baudot-Codes ist, daß die einzelnen Zeichen in sich nicht eindeutig sind; so kann eine bestimmte 5-bit-Folge sowohl einen Buchstaben als auch eine Ziffer oder ein Zeichen darstellen – je nachdem, ob irgendwann vorher einmal ein Zeichen, „Buchstabe" oder „Ziffer", gesendet wurde. Der Baudot-Zeichenvorrat ist also zunächst einmal in zwei Teilbereiche getrennt, nämlich die Buchstaben von A bis Z und die Ziffern und Zeichen. Wurde z. B. als letztes Zeichen ein Buchstabe gedruckt, so muß vor der Ausgabe einer Ziffer oder

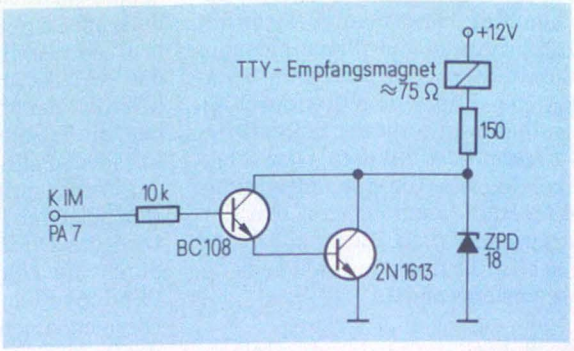
```
UNTERPROGRAMM ZUR BAUDOT-AUSGABE
0352 86 F5      STX F5    X RETTEN
0354 A2 80      LDX #80   PA 7 =
0356 8E 01 17   STX 1701  AUSGANG
0359 C9 0D      CMP #0D   C.R.
035B D0 04      BNE 0361
035D A9 08      LDA #08
035F D0 0E      BNE 036F
0361 C9 0A      CMP #0A   L.F.
0363 D0 04      BNE 0369
0365 A9 02      LDA #02
0367 D0 06      BNE 036F
0369 29 3F      AND #3F   6 BIT ASCII
036B AA         TAX       ALS INDEX
036C BD C0 03   LDA 03C0,X
036F 85 F4      STA F4
0371 29 20      AND #20   MODUS-TEST
0373 C5 F3      CMP F3    (BU./ZI.)
0375 F0 0E      BEQ 0385
0377 85 F3      STA F3    NEUER MODUS
0379 A8         TAY       NULL-TEST
037A F0 04      BEQ 0380
037C A9 1B      LDA #1B
037E D0 02      BNE 0382  MODUS-
0380 A9 1F      LDA #1F   UMSCHALTUNG
0382 20 8D 03   JSR 038D  AUSGEBEN
0385 A5 F4      LDA F4    UND ZEICHEN
0387 20 8D 03   JSR 038D DRUCKEN
038A A6 F5      LDX F5
038C 60         RTS
038D A2 7F      LDX #7F   STARTBIT
038F 8E 00 17   STX 1700
0392 20 B4 03   JSR 03B4
0395 A0 04      LDY #04   5 BITS
0397 4A         LSR
0398 90 04      BCC 039E
039A A2 FF      LDX #FF   '1'
039C D0 02      BNE 03A0
039E A2 7F      LDX #7F   '0'
03A0 8E 00 17   STX 1700
03A3 20 B4 03   JSR 03B4
03A6 88         DEY
03A7 10 EE      BPL 0397
03A9 A2 FF      LDX #FF   STOPBIT
03AB 8E 00 17   STX 1700  (LAENGE =
03AE A2 1E      LDX #1E   1,5 BIT)
03B0 20 B6 03   JSR 03B6
03B3 60         RTS
03B4 A2 14      LDX #14   VERZOEG.1 BIT
03B6 8E 07 17   STX 1707
03B9 2C 07 17   BIT 1707
03BC 10 FB      BPL 03B9
03BE 60         RTS
03BF 00                   ENDE
```

Bild 1. Unterprogramm zur Ausgabe eines im Akku stehenden ASCII-Zeichens auf einem Baudot-Fernschreiber

```
TABELLE 6-BIT-ASCII ZU BAUDOT

03C0 00 03 19 0E 09 01 0D 1A
03C8 14 06 0B 0F 12 1C 0C 18
03D0 16 17 0A 05 10 07 1E 13
03D8 1D 15 11 2D 00 3A 00 00
03E0 24 34 00 29 00 24 31 25
03E8 2F 32 39 31 2C 23 3C 3D
03F0 36 37 33 21 2A 30 35 27
03F8 26 38 2E 2B 24 3E 24 39
0400 ENDE
```

◀ Bild 2. Tabelle zur Umwandlung eines 6-bit-ASCII- in ein 5-bit-Baudot-Zeichen

Bild 3. ▶ Anschluß eines Fernschreiber-Druckers an den Mikrocomputer KIM-1




```
ASCII-EINGABE
BAUDOT-AUSGABE

0000 20 5A 1E    JSR 1E5A  GETCH(KIM)
0003 20 52 03    JSR 0352  BAUDOT-AUSG.
0006 4C 00 00    JMP 0000  NOCHMAL
```

◀ Bild 4. Einfaches Anwendungsbeispiel: Programm zum Schreiben auf dem TTY-Drucker mit einer ASCII-Tastatur

eines Satzzeichens erst einmal das Zeichen „ZI" ausgegeben werden.

Ganz anders beim ASCII-Code (American Standard Code for Information Interchange): Jedes Zeichen ist für sich eindeutig definiert; außerdem handelt es sich bei diesem Code normalerweise um ein 7-bit-Format, das $2^7 = 128$ verschiedene Zeichen darstellen kann, wesentlich mehr, als das beim Baudot-Code ($2 \cdot 2^5 = 64$ Zeichen) möglich ist. So sind im ASCII-Code auch Kleinbuchstaben sowie eine ganze Reihe von Steuerzeichen möglich. Aus diesen Gründen setzte sich ASCII in der Mikrocomputertechnik praktisch hundertprozentig durch.

Programmieren kostet Zeit...

...und Zeit ist Geld. Und deswegen honoriert die FUNKSCHAU jede in der Mikrocomputer-Rubrik veröffentlichte Druckseite mit 200 DM. Dafür erwarten wir aber auch etwas: zum Beispiel einen reproduktionsfähigen Programmausdruck (zur Not genügt auch ein sauber mit der Schreibmaschine abgetipptes Programm) und einen erklärenden Text, was das Programm überhaupt tut. Falls Sie keinen Drucker, aber einen KIM-1 oder einen TI-59 besitzen, genügt es auch, wenn Sie uns Ihr Programm auf Kassette bzw. Magnetkarte übersenden. Und noch eine Möglichkeit: Sie können uns Ihr Programm als Auflistung auch mit dem FSK-Modem auf Kassette überspielen, das in Heft 15/1978 und im Hobbycomputer-Sonderheft des Franzis-Verlages beschrieben war – aber bitte mit 110 Baud und inklusive CR, LF und Leerräumen in ASCII!

Das Programm

Aufgabe des in *Bild 1* dargestellten Unterprogramms ist es, ein im Akku stehendes ASCII-Zeichen auf einem Baudot-Fernschreiber auszudrucken. Um Platz für die notwendige Umwandlungstabelle (*Bild 2*) zu sparen, werden nur 6 bit des 7-bit-ASCII-Zeichens als Tabellenindex verwendet; die beiden Steuerzeichen CR (Wagenrücklauf) und LF (neue Zeile) werden vom Programm getrennt decodiert, da sie bei der Einschränkung auf 6 bit verlorengingen. Der 6-bit-Maskierung fallen auch die Kleinbuchstaben zum Opfer; sie können aber auf dem Fernschreiber ohnehin nicht ausgegeben werden.

Das „Haupt-Unterprogramm" ruft zwei weitere Unterprogramme auf, nämlich eines zur Ausgabe eines im Akku stehenden Baudot-Zeichens (Adresse 038D) und eines zur Vergrößerung um eine der jeweiligen Baud-Rate entsprechende Bit-Dauer, hier für 50 Baud dimensioniert.

Es wurde darauf geachtet, daß die Wirkung dieses Programms auf die Prozessor-Register genau identisch ist mit dem im KIM-Monitor-ROM stehenden Unterprogramm „OUTCH", d. h. das X-Register wird „gerettet", Y ist nach der Rückkehr vom Unterprogramm hexadezimal FF, und der Akkuinhalt ist verloren.

Es muß noch erwähnt werden, daß die Tabelle in Bild 2 außer der 5-bit-Baudot-Information als sechstes Bit noch die Aussage enthält, ob es sich um den Buchstaben- oder um den Ziffernmodus handelt. Ändert sich der Modus gegenüber dem zuletzt ausgegebenen Zeichen, so wird ein entsprechendes Umschaltzeichen an den Fernschreiber ausgegeben. Das Programm benutzt die Zero-Page-Adressen 00F3 (Modus), 00F4 (Baudot-Zeichen) und 00F5 (X-Register).

Notwendige Hardware

Die Timer-Adressen (1707) und die Adressen der I/O-Register beziehen sich auf den Mikrocomputer KIM-1. Das Baudot-Zeichen wird am Anschluß PA 7 ausgegeben; das Programm gestattet es aber, auch einen beliebigen anderen Port als Ausgang zu verwenden. Zum Anschluß des Baudot-Fernschreibers genügt eine Schaltung nach *Bild 3*. Die Z-Diode dient dazu, Schaltspitzen zu dämpfen, die durch die Selbstinduktion des Empfangsmagneten auftreten.

Bild 4 zeigt schließlich ein einfaches Programm, das das direkte Schreiben auf dem Drucker des Fernschreibers mit einem ASCII-Terminal gestattet; es besteht nur aus zwei Unterprogramm-Aufrufen und einem Rücksprungbefehl. Sollte sich herausstellen, daß die Fernschreibergeschwindigkeit nicht 50 Baud beträgt, so sind die hexadezimalen Inhalte der Zellen 03AF und 03B5 zu ändern.

Ein Verlegen des Baudot-Ausgabe-Unterprogramms in eine andere „Page", also in eine andere Speicherseite, ist sehr einfach: Man braucht nur bei allen 3-Byte-Befehlen, deren drittes Byte 03 lautet, dieses Byte auf die neue Page ändern. *Herwig Feichtinger*

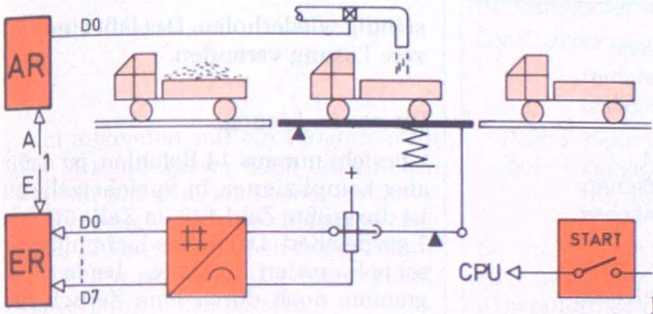
Hans Werner

Mikrocomputer

Eine kleine Einführung

8. Teil

Einfache Waage



Eine einseitig gelagerte Plattform steht auf einer Feder und lenkt bei Belastung den Schleifer eines Potentiometers aus. Die Schleiferspannung soll proportional zur Gewichtsbelastung sein. Sie wird durch einen Analog-Digitalumsetzer in die Binärpegel D0...D7 umgesetzt, die am Eingangsregister des Mikrocomputers liegen.

Richtige Waagen in dieser Größenordnung sind, statt mit der Feder-Potentiometerkombination, mit Wägezellen ausgerüstet. Für die prinzipielle Betrachtung ist das gleichgültig. In beiden Fällen ist das Ausgangssignal eine Spannung, proportional zur Belastung.

Problem

Mit einem Füllrüssel sollen automatisch Lastwagen beladen werden.

Bedingungen

1. Alle Lastwagen sollen mit der gleichen Materialmenge beladen werden.
2. Das Eigengewicht der Laster (Tara) darf keinen Einfluß auf die Messung haben.

Ablauf

Das Lastauto fährt auf eine Plattform, der Fahrer drückt auf den Knopf START und der Absperrschieber öffnet sich automatisch. Nach dem Erreichen der gewünschten Materialmenge schließt der Schieber.

Festlegung

Der Absperrschieber wird durch das Ausgaberegister gesteuert.
D0=0: Schieber schließt
D0=1: Schieber öffnet

Die 0 soll in Zelle 27, die 1 in 28 und der Grenzwert (Füllmenge) in 26 gespeichert sein. AR,A und ER,E werden fest auf 1 gelegt.

```
                    Start
0   LA 31   Tara T (leeres Auto)
            in den AC laden
1   SP 30   Tara T in Zelle 30
            speichern

2   LA 28   Zahl 1 in den AC
            laden
3   SP 31   Zahl 1 ausgeben:
            Absperrschieber auf

4   LA 31   Brutto B (Auto +
            Material) in AC laden
5   SU 30   Tara subtrahieren:
            Netto N = B-T

6   SP 29   Nettogewicht in
            Zelle 29 speichern
7   LA 26   Grenzwert in den AC
            laden
8   SU 29   Nettogewicht
            subtrahieren

9   JP  1   Sprung, wenn Inhalt
            AC ungleich 0

10  LA 27   Grenzwert erreicht:
            0 in AC laden
11  SP 31   Zahl 0 ausgeben:
            Absperrschieber zu
12  ST      Stop! Warten auf den
            nächsten Start
```

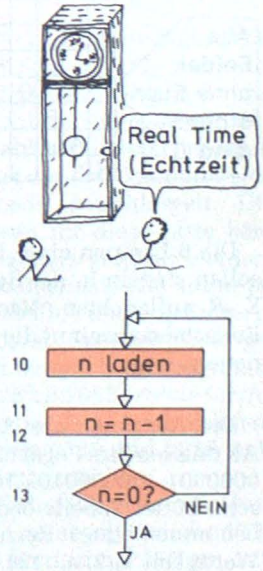
Kritik am Programm

1. Beim ersten Durchlauf des Programms ist die Schieberstellung nicht eindeutig definiert. Abhilfe: Programm mit den Befehlen LA 27, SP 31 beginnen.

2. Der Grenzwert darf nicht durch eine einzelne Zahl, sondern muß durch eine Toleranzzone definiert werden. Nur dann wird vermieden, daß evtl. die richtige Füllmenge vor oder nach Befehl 4 erreicht und somit der Sprungbefehl immer ausgeführt wird.

Zeit-Schleife

Läßt man durch eine Programmschleife mehrmals die gleichen Befehle bearbeiten, vervielfacht sich die Laufzeit.

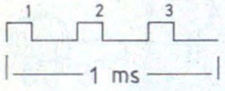


Auf diese Art können z. B. Wartezeiten oder Impulse bestimmter Dauer erzeugt werden. Mit der wirklichen Uhrzeit (Echtzeit) hat das natürlich nicht unbedingt zu tun.

Ein Speicherzelle wird als Zähler benutzt, die Anzahl n der gewünschten Programmdurchläufe gespeichert und bei jedem Durchlauf 1 substrahiert. Nach n Durchläufen ist die Zeitschleife abgearbeitet.

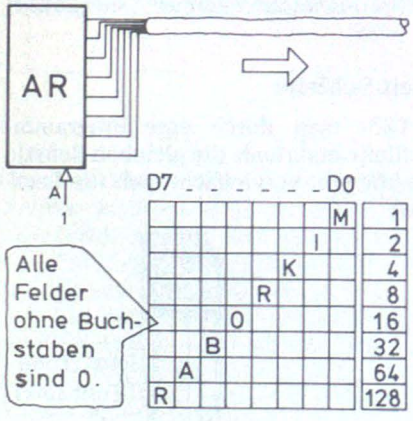
```
 8  LA 28   Zahl n in        Speicher-
 9  SP 30   den Akku         belegung:
             laden            (28) =
10  LA 30   n in den         Anzahl der
             Akku laden       Durchläufe
11  SU 29   1 subtra-          n
12  SP 30   hieren           (29) = Zahl 1
13  SP 9    Ist n = 0?
```

Zahlenbeispiel: n = 100
Befehlszyklus = 1 ms



Die Schleife besteht aus 4 Befehlen, die 100 mal durchlaufen werden. Das ergibt eine Gesamtzeit von 400 ms.

Lauflicht



Die 8 Lampen einer Barbeleuchtung sollen einzeln in der Reihenfolge M, I, K...R aufleuchten. Nach dem letzten Buchstaben beginnt die Laufschrift von neuem.

Die Ausgänge des Ausgaberegisters AR müssen ihre Pegel nach dem Muster 0000001, 00000010...10000000 wechseln. In der Tabelle bedeuten Buchstaben immer eine 1. Rechts ist die binäre Wertigkeit 1, 2, 4...128 aufgetragen.

Der Befehlssatz des Computermodells hat keinen speziellen Schiebebefehl, mit dem eine 1 von Platz D0 bis D7 geschoben werden kann. Der Befehl kann leicht durch fortlaufende Additionen ersetzt werden, denn die Zahlenwerte verdoppeln sich ständig:

```
1
1+1 = 2
2+2 = 4
4+4 = 8
bis
64+64 = 128
```

```
LA 30    1 laden
SP 29    1 speichern
SP 31    1 ausgeben

AD 29    1 + 1
SP 29    2 speichern
SP 31    2 ausgeben

AD 29    2 + 2
SP 29    4 speichern
SP 31    4 ausgeben
 .
 .
 .
AD 29    64 + 64
SP 29    128 speichern
SP 31    128 ausgeben

JP       Rücksprung
```

Zuerst eine einfache Lösung des Problems. In Zelle 30 ist eine 1 gespeichert. Das Programm besteht aus 3·8+1 = 25 Befehlen, die sich zum großen Teil

ständig wiederholen. Das läßt eine kürzere Lösung vermuten.

Die zweite Lösung...

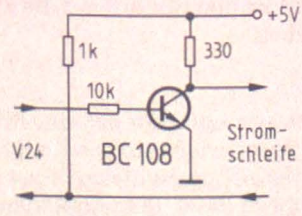
...besteht nur aus 14 Befehlen, ist dafür aber komplizierter. In Speicherzelle 29 ist die größte Zahl 128, in Zelle 30 eine 1 gespeichert. Damit das Licht nicht zu schnell rotiert, können beide Programme noch durch eine Zeitschleife ergänzt werden.

```
 0  LA 30   Zahl 1 in den Akku laden
 1  SP 28   Zahl 1 in 28 speichern
 2  SU 28
 3  SP 27   Zahl 0 in 27 speichern

 4  LA 27   Inhalt von 27 in Akku
 5  AD 28   1, 2, 4...64 addieren
 6  SP 27
 7  SP 28   Ergebnis speichern
 8  SP 31   1, 2, 4...128 ausgeben

 9  LA 29   Zahl 128 in Akku laden
10  SU 27   1, 2, 4...128 subtrahieren
11  JP  3   bei (AC) ungleich 0,
             nach 4
12  LA 30   damit (AC) ungleich 0
13  JP  0   nach 1 springen
```

(Fortsetzung folgt)

So einfach ist es...

...eine RS-232-Schnittstelle mit einer 20-mA-Stromschleife zu verbinden, z. B. ein CT-64-Terminal mit dem Mikrocomputer KIM-1. Zwar ist die Pegelanpassung nicht ganz normgerecht – aber es funktioniert.



Walter Krautzig

Ein Musik-Programm für den Computer

Was fängt man mit einem Computer an? Das fragt sich so mancher, der sich beispielsweise den 8080-Mikrocomputer nach der Artikelserie von Horst Pelka in der FUNKSCHAU (1977/78) nachgebaut hat. Gemeinhin stellt man sich unter solch einem Gesellen – wenn man das Ding personifiziert – einen seelen-, humor- und gefühllosen Rechenroboter vor. Das ist jedoch, wie bei uns Menschen auch, lediglich eine Sache der Erziehung – sprich: Programmierung.

Im folgenden soll ein Programm für den Mikroprozessor 8080 beschrieben werden, mit dem man seine erstaunlichen musischen Eigenschaften wecken kann. Er kann damit beliebige Melodien, bis hin zu anspruchsvollen, klassischen Solostücken spielen.

Theoretische Grundlagen

Der Grundgedanke der Musikerzeugung durch den Computer ist denkbar einfach: Man veranlaßt ihn, per Programm in abwechselnder Reihenfolge High- und Low-Impulse definierter Länge an den niederwertigsten Ausgang (D_0) eines Ausgabekanals zu geben. Direkt über einen Kopfhörer oder über einen kleinen Verstärker am Lautsprecher kann man die Impulsfolge als Ton abhören. Da die zusammengesetzten Impulse eine Rechteckschwingung bilden, ist der Ton recht obertonreich und klingt angenehm.

Per Programm muß dem Computer die Schwingungsdauer (Tonhöhe) und die Tonlänge (Notenlänge) angegeben werden. Beide Zeiten müssen gleichzeitig in zwei ineinander verschlungenen Verzögerungsschleifen abgezählt werden. Die Schwingungsdauer (T) ist umgekehrt proportional der Tonfrequenz (f):

$$T = \frac{1}{f}$$

Wir brauchen die Schwingungsdauer der Halbschwingung, also

$$\frac{T}{2} = \frac{1}{2f}$$

Um diese Halbschwingungsdauer abzuzählen, benutzen wir die kleinstmögliche Verzögerungsschleife, nämlich das Dekrementieren eines Registers bis auf Null:

```
DCR r   dekrementierte Register
JNZ     springe zurück, wenn Ergeb-
        nis nicht null
```

Da für die Ausführung jeden Befehls eine bestimmte Anzahl Taktperioden (Operationsschritte) gebraucht werden und die Taktfrequenz durch den an den Taktgenerator 8224 angeschlossenen Quarz bestimmt ist, läßt sich die Dauer eines Schleifendurchgangs exakt ermitteln.

Die Eigenfrequenz des von Horst Pelka vorgeschlagenen Quarzes beträgt 18,432 MHz. Die Taktfrequenz des Prozessors dann 18,432 : 9 = 2,048 MHz. Daraus folgt die Taktdauer oder Dauer eines Operationsschrittes

$$T_p = \frac{1}{2{,}048 \cdot 10^6} = 0{,}4883 \cdot 10^{-6}\ \text{s}$$

Es werden für DCR r 5 Operationsschritte und für JNZ 10 Schritte, insgesamt also 15 gebraucht.

Ein Schleifendurchgang dauert daher $15 \cdot 0{,}4883 \cdot 10^{-6} = 7{,}324 \cdot 10^{-6}$ s, also 7,324 millionstel Sekunden. Um z. B. den Kammerton a mit einer Frequenz von 440 Hz zu erzeugen, sind

$$\frac{1}{2 \cdot 440 \cdot 7{,}324 \cdot 10^{-6}} = 155{,}2$$

Schleifendurchgänge erforderlich. Nun sind jeweils nach jedem Beenden dieser Tonhöhenschleife eine Anzahl Befehle zu durchlaufen, nämlich das Dekrementieren und Zurückspringen für das Abzählen der Notenlänge sowie das Umschalten des Impulses. In der Programmfolge sind für diesen Programmteil die Operationsschritte angegeben. Diese Zeitverzögerung soll mit durchschnittlich 60 Operationsschritten = ca. 29,3 µs angenommen werden, weil ja nicht alle Schleifen bei jedem Impulswechsel durchfahren werden. Um diese Zeit zu kompensieren, werden die Durchgänge der Tonhöhenschleife um etwa 4 (= 29,3 : 7,324) vermindert. „Etwa" deswegen, weil ja nur ganzzahlige Durchgänge möglich sind. Es muß also die exakt berechnete Zahl der Durchgänge auf ganze Zahlen auf- oder abgerundet werden. Die dadurch bewirkte „Verstimmung" ist aber normalerweise nicht wahrnehmbar.

Aus Platzgründen wurde die Programm-Auflistung für den 8080 hier nicht abgedruckt. Interessenten erhalten sie gegen Einsenden eines Freiumschlages von der FUNKSCHAU-Redaktion, Postfach 37 01 20, 8000 München 37.

In der *Tabelle 1* ist für die Töne c^1 bis f^3, also $2\,^1/_2$ Oktaven, die Zahl der Schleifendurchgänge in Dezimal- und Hexadezimalzahlen angegeben. Die Hexadezimalzahl ist der Speicherinhalt in der Liste, aus der der Computer sich den Melodiecode entschlüsselt. Die Speicheradressen für diese Liste können natürlich anders gewählt werden. Die letzte Hexadezimalstelle sollte jedoch mit der Tabelle 1 übereinstimmen. Wird ein anderer Speicherblock gewählt, müssen lediglich im Programm die Befehle 052E und 0531 sowie für die Decodierung von Ton- und Pausenlänge die Befehle 0535 und 053A entsprechend geändert werden.

Es sei vermerkt, daß bei dem tiefsten Ton (c^1) etwas „gemogelt" wurde: FFH Schleifen sind für dezimal 257 um 2 zu

wenig. Aber dieser Ton muß von demjenigen, der das Programm noch erweitern will und dafür den 5-bit-Code 02 braucht, wie noch erklärt werden wird, herausgenommen werden.

Die *Tabelle 2* zeigt die entsprechende Speicherbelegung für das Decodieren der Noten- und Pausenlängen.

Der Ton, der am Ausgang D_0 zu hören ist, erklingt an D_1 um eine Oktave, an D_2 um zwei Oktaven tiefer. Durch einen einfachen Umschalter, für den als Hardware lediglich 3 Nandgatter mit 2 Eingängen (z. B. 7400) nötig sind, kann man den Ausgang zwischen D_0 und D_2 umschalten und den Tonumfang dadurch auf $4\,^1/_2$ Oktaven erweitern (*Bild 1*).

Tabelle 2:
Decodierliste und Speicherplatzbelegung für Noten- und Pausenlängenzahlen

Notenlängen dezim.	(Pausen-) zahl hexadez.	Speicher Adresse	Note oder Pause	3-bit-Code
04	04	04C1	1/16	01
08	08	04C2	1/8	02
12	0C	04C3	3/16	03
16	10	04C4	1/4	04
24	18	04C5	3/8	05
32	20	04C6	1/2	06
48	30	04C7	3/4	07

Die Melodieliste

Die Töne, Pausen und besonderen Spielanweisungen werden dem Computer als Folge von 8-bit-Speicherworten eingegeben, die durch das eigentliche Ablaufprogramm gelesen und decodiert werden.

Das erste Wort dieser Liste ist stets die Angabe der Anzahl der Durchgänge (Strophen), die als Hexadezimalzahl eingegeben wird. Es folgt als 2. und 3. Wort die Anweisung für das gewünschte Tempo. Es besteht aus 0C, dem Code für „Tempo" und nachfolgend der das Tempo bestimmenden Zahl. 0C 10 bedeutet, daß pro Minute etwa 120-$^1/_4$-Noten gespielt werden. Zu größeren Zahlen hin wird das Tempo langsamer; es kann im Verlauf des Musikstücks durch Einfügung neuer Tempoanweisungen beliebig oft gewechselt werden.

Nach diesen ersten 3 Listenworten folgen entsprechend der Melodie die Codeworte für die einzelnen Noten und Spielanweisungen.

Das 8-bit-Speicherwort setzt sich zusammen aus den oberen 5 Bits (D_3...D_7) als Code für die Tonhöhe und den unteren 3 Bits (D_0...D_2) als Code für die Noten- bzw. Pausenlänge. Ausgenommen sind der 5-bit-Code 0000, der „Pause" bedeutet, wobei die nachfolgenden 3 Bits die Pausenlänge angeben, und der Code 00001, den der Computer als Spielanweisung interpretiert. Die Art der Anweisung ist in den folgenden 3 Bits enthalten. In *Tabelle 3* sind die 8 möglichen Spielanweisungen aufgeführt und erklärt. Der Code 00010 ist im Programm für zusätzliche Spielanweisungen freigehalten (Dynamik, Klangformung), um die sich der Anwender das Programm selbst erweitern kann. Der Sprungbefehl 052B, welcher diese Decodierung jetzt überspringt, braucht dann nur geändert zu werden. Der Ton c^1 (bzw. C im Baß), der jetzt durch 00010 codiert wird, fällt dann weg.

Die Angaben in der letzten Spalte der Tabellen 1 und 2 sollen die Zusammensetzung der 5 + 3 bit zu der hexadezimalen Schreibweise von 4 + 4 bit erleichtern. Hat der Code in Tabelle 1 als Einer eine Null, so bedeutet das, daß der 5-bit-Code als unterstes Bit 0 hat. Es braucht dann der Zehnerstelle aus Ta-

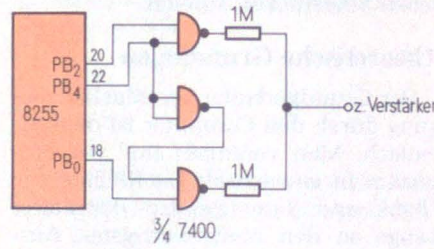


Bild 1. Notwendige Hardware für die Baß-Umschaltung

Tabelle 1: Umrechnung der Tonfrequenzen auf Tonhöhenzahlen und Belegung der Decodierliste

Ton	Frequenz	Schleifenzahl	Tonlängenzahl dezim.	hexadez.	Speicher Adresse	Ton	5-bit-Code
c^1	261,6	261	257	FF	04A2	c^1	10
cis	277,2	246,3	242	F2	04A3	cis/des	1+
d	293,7	232,4	228	E4	04A4	d	20
dis	311,1	219,4	215	D7	04A5	dis/es	2+
e	329,6	207,1	203	CB	04A6	e	30
f	349,2	195,5	192	C0	04A7	f	3+
fis	370	184,5	181	B5	04A8	fis/ges	40
g	392	174,2	170	AA	04A9	g	4+
gis	415,3	164,4	160	A0	04AA	gis/as	50
a	440	155,2	151	97	04AB	a	5+
ais	466,2	146,4	142	8E	04AC	ais/b	60
h	493,9	138,2	134	86	04AD	h	6+
c^2	523,2	130,5	127	7F	04AE	c^2	70
cis	554,3	123,2	119	77	04AF	cis/des	7+
d	587,3	116,2	112	70	04B0	d	80
dis	622,2	109,7	106	6A	04B1	dis/es	8+
e	659,3	103,5	100	64	04B2	e	90
f	698,5	97,7	94	5E	04B3	f	9+
fis	740	92,3	88	58	04B4	fis/ges	A0
g	784	87,1	83	53	04B5	g	A+
gis	830,7	82,2	78	4E	04B6	gis/as	B0
a	880	77,6	74	4A	04B7	a	B+
ais	932,3	73,2	69	45	04B8	ais/b	C0
h	987,8	69,1	65	41	04B9	h	C+
c^3	1046,5	65,2	61	3D	04BA	c^3	D0
cis	1108,7	61,6	58	3A	04BB	cis/des	D+
d	1174,7	58,2	54	36	04BC	d	E0
dis	1244,5	54,9	51	33	04BD	dis/es	E+
e	1318,5	51,8	48	30	04BE	e	F0
f	1396,9	48,9	45	2D	04BF	f	F+

belle 1 nur die Einerstelle aus Tabelle 2 angehängt zu werden, um den 8-bit-Code zu erhalten. Steht als Einer in Tabelle 1 ein +, bedeutet das eine 1 als unterstes der 5 Bits. Dann muß dem 3-bit-Code hexadezimal 8 hinzuaddiert werden. Ein Beispiel:

$^1/_4$-Note e^1 : 30 + 04 = 34; $^1/_4$-Note f^1 : 3+ + 04 = 3C. Das letzte Speicherwort der Melodieliste muß immer OF lauten.

Das Programm

Man sollte ein umfangreicheres Programm immer in möglichst kleine Teilprogramme gliedern. Es wird dann überschaubarer, und die Fehlersuche wird erleichtert (*Bild 2*).

Die einzelnen Programme sind hier lediglich in Form von Flußdiagrammen dargestellt. Dem Anwender, der das Programm direkt übernehmen kann (notwendiges System: 8080-Konfiguration wie in der FUNKSCHAU-Serie „Praxis mit Mikroprozessoren"), sendet die Redaktion gerne eine komplette Auflistung des Quellen- und Objektcodes zu.

In *Tabelle 4* sind die einzelnen RAM-Speicherplätze aufgelistet. Im Computer des Verfassers steht das Programm im CMOS-RAM, der von 0400...07FF reicht. Der Arbeitsspeicher mit 1 KByte RAM reicht von 8000...83FF.

Die Anfangsadresse der Melodieliste muß „von Hand" in das Speicherzellenpaar 8000/8001 geschrieben werden, und zwar das niederwertigste Byte nach 8000.

Leitprogramm „Legato"

Das Registerpaar HL wird als Zeiger auf die Anfangsadresse der Melodieliste gesetzt. Die Strophenzahl wird gelesen, und HL wird umregistriert auf die Adresse 83D6, wo die Strophenzahl gespeichert wird. In 056C wird HL auf 83D7 erhöht. Diese Speicherstelle wird im Programm mehrere Male gebraucht. Durch die hier angewandte indizierte Adressierung über HL werden Programmspeicherstellen gespart. Das Re-

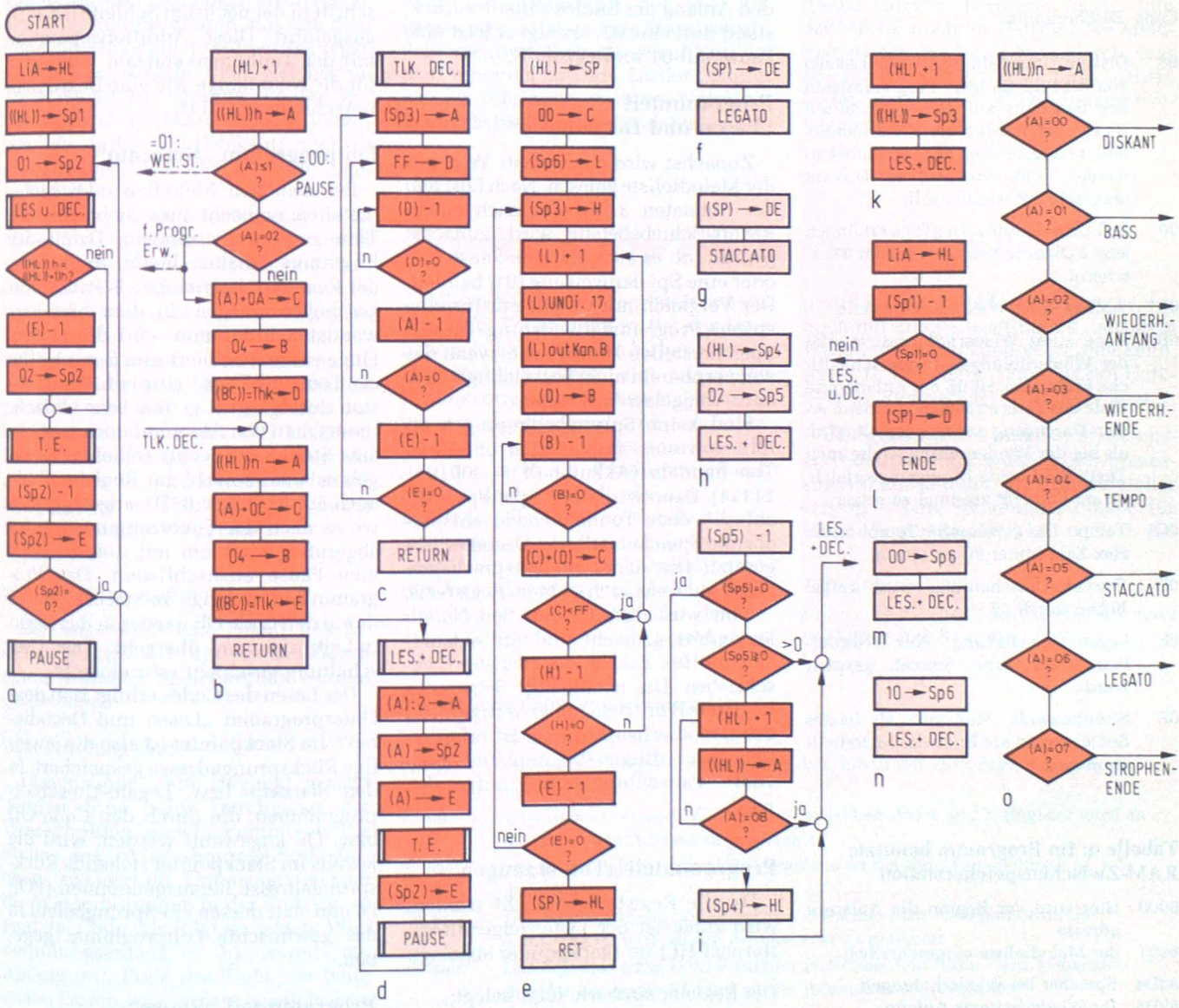


Bild 2. Flußdiagramme der einzelnen Teilprogramme: Legato (a), Lesen und Decodieren (b), Pause (c), Staccato (d), Tonerzeugung (e), Umschalten auf Legato (f), Umschalten auf Staccato (g), Wiederholungsanfang (h), Wiederholungsende (i), Tempo (k), Strophenende (l), Diskant-Umschaltung (m), Baß-Umschaltung (n) und Weichenstellung (o). Die Arbeitsspeicher enthalten die Information für die Strophenzahl (Sp 1), für verschiedene Konstanten (Sp 2), den Tempo-Koeffizienten (Sp 3), den Listenstand bei Wiederholungsbeginn (Sp 4), eine Konstante im Wiederholungsprogramm (Sp 5) und das Impulswort (Sp 6). LiA ist die Anfangsadresse der Melodieliste, und mit SP ist der letzte Inhalt des Kellerspeichers (Stack) bezeichnet. Index h bezeichnet die 5 höherwertigen Bits (D7...D3) des vom Register HL adressierten Listenworts, Index n entsprechend die niederwertigen Bits (D2...D0)

gisterpaar HL kann durch die XTHL-Operation abwechselnd auf die Melodieliste und die Speicherstelle 83D7 gesetzt werden.

Bei 056F steht HL wieder auf der Anfangsadresse der Melodieliste. Es wird jetzt das Lese- und Decodierprogramm aufgerufen. Zunächst soll aber das Leitprogramm weiter besprochen werden: Der Rücksprung erfolgt erst, wenn der erste Ton gelesen und decodiert worden ist. Das Lesen und Decodieren geht nun so schnell, daß die Töne ohne hörbare Pause ineinander übergehen. Kommen in der Melodie mehrere Töne der gleichen Höhe hintereinander vor, so würden diese ineinander verlaufen und als ein langer Ton zu hören sein. Daher muß der Computer prüfen, ob die Tonhöhe des gerade decodierten sich von dem nächstfolgenden unterscheidet. Wenn nicht, muß er den ersten Ton etwas kürzer spielen und die gekürzte Zeit als Pause anhängen. Dazu vergleicht er durch eine Exclusiv-Oder-Funktion und Abtrennen der untersten 3 bit, ob die beiden Tonhöhen gleich sind. Wenn ja, wird die Notenlängenzahl im Register E um 1 vermindert und in den Speicher 83D7 statt der dort stehenden 1 eine 2 geschrieben. Nach Durchlaufen des Tonerzeugungsprogramms wird Speicherstelle 83D7 dekrementiert. Stand dort 01, springt er an den Anfang der Endlosschleife zurück; stand dort eine 02, erzeugt er jetzt eine Pause mit 01 im Register E.

Tabelle 3: Die acht Spielanweisungen

Code	Beschreibung
08	Diskant-Umschaltung = Rücksetzen von Bit PB_4. Da beim Programmieren des Bausteins alle Bits automatisch rückgesetzt werden, braucht dieser Befehl im allgemeinen nur verwendet zu werden, wenn von Baß auf Diskant umgeschaltet werden soll.
09	Baß-Umschaltung. Die Töne erklingen jetzt 2 Oktaven tiefer, bis wieder 08 erscheint.
0A	Anfang eines Wiederholungsteils.
0B	Ende eines Wiederholungsteils. Hat der Wiederholungsteil unterschiedliche Endungen, ist 0B am Anfang und Ende des Teils zu setzen, der beim ersten Durchgang anders gespielt wird, als bei der Wiederholung (siehe auch Text). Sind beide Durchgänge gleich, ist am Ende 0B zweimal zu setzen.
0C	Tempo. Das gewünschte Tempo ist als Hex-Zahl hinter 0C zu setzen.
0D	Staccato-Umschaltung, wird aufgehoben durch 0E.
0E	Legato-Umschaltung. Nur erforderlich, wenn vorher Staccato gespielt wurde.
0F	Strophenende. Muß stets als letztes Speicherwort am Ende der Melodieliste stehen.

Tabelle 4: Im Programm benutzte RAM-Zwischenspeicherstellen

8000	Hier wird vor Beginn die Anfangsadresse
8001	der Melodieliste eingeschrieben
83D4	Speicher bei Wiederholungen
83D5	für Wiederholungs-Anfang
83D6	Strophenzahl
83D7	intern. Zwischenspeicher
83D8	Speicher für Ausgabe-Byte
83F4	Zwischenspeicher bei Wiederholung
83F5	Tempo

Programmteil „Lesen und Decodieren"

Zunächst wird das nächste Wort aus der Melodieliste gelesen. Nach Löschen der untersten 3 bit und nach einem Rechts-Schiebebefehl wird zunächst geprüft, ob es sich um eine Pause (00) oder eine Spielanweisung (01) handelt. Der Vergleich mit 02 gilt nur für eine spätere Programmerweiterung. Diese 5 Speicherstellen können also, wenn das von vornherein nicht beabsichtigt wird, auch weggelassen werden.

Sind keine Sprungbedingungen erfüllt gewesen, muß es sich um einen Ton handeln (Akkuinhalt = 00010 – 11111). Dann wird das Registerpaar BC auf die dem Tonhöhencode entsprechende Speicherstelle der Decodierliste gesetzt. Die dem Ton entsprechende Hex-Zahl steht nach 0534 im Register D.

Nun wird dasselbe mit den Notenlängenbits gemacht und die entsprechende Hex-Zahl in das Register E geschrieben. Das nochmalige Setzen des Registers B mit dem höherwertigen Teil der Decodierlistenadresse ist erforderlich, weil dieser Programmteil auch vom Pausenprogramm aufgerufen wird.

Programmteil „Ton erzeugen"

Da alle Register gebraucht werden, wird zunächst der Listenzeiger (Registerpaar HL) im Stackpointer abgelegt.

Die Register sind wie folgt belegt:

B: Dekrementierregister für die Tonhöhenschleife
C: Summenregister für die Addition der Tonhöhenzahl
D: Speicher für die Tonhöhenzahl
E: Dekrementierregister für die decodierte Notenlängenzahl
H: Dekrementierregister für den Tempo-Koeffizienten
L: Speicherregister für den Tonimpuls.

Im übrigen besteht dieser Programmteil aus Verzögerungsschleifen, deren Funktionsweise als bekannt vorausgesetzt werden kann. Da hier zwei Zeitverzögerungen gleichzeitig arbeiten müssen, ist es wichtig, wie beide miteinander verbunden werden. Nach jedem Beenden der Tonhöhenschleife wird die Tonhöhenkennzahl in das Register C addiert. Bei Überlauf über FFH, also wenn nach einer Addition das Carrybit gesetzt ist, wird ein Dekrementierschritt in der nächsten Schleife (DCR H) ausgeführt. Diese Additionsoperation mit der Tonhöhenkennzahl wirkt also für die Notenlänge wie eine Dekrementierschleife mit FFH.

Leitprogramm „Staccato"

Bei manchen Melodien oder Melodieteilen wünscht man sich, daß die Töne nur kurz erklingen. Damit der Rhythmus erhalten bleibt, muß dann der Rest der Notenlänge als Pause eingeschoben werden. In dem hier verwendeten Programm wird die Notenlänge einfach halbiert und je zur Hälfte als Ton und Pause gespielt. Die Division durch 2 geht ja dual sehr einfach, indem man den Akkumulatorinhalt um eine Stelle nach rechts schiebt. Das Ergebnis wird sowohl im Register E als auch im Speicher 83D7 abgelegt, von wo es nach der Tonerzeugung wieder abgerufen wird, um mit diesem Wert eine Pause anzuschließen. Das Programm läuft so lange im Staccato, bis es durch den Code OE wieder in das Legato-Leitprogramm übergeht. Die Umschaltung geschieht sehr einfach:

Das Lesen des Codes erfolgt ja in dem Unterprogramm „Lesen und Decodieren". Im Stackpointer ist also die jeweilige Rücksprungadresse gespeichert. In den Staccato- bzw. Legato-Umschaltprogrammen, die durch den Code OD bzw. OE angewählt werden, wird die jeweils im Stackpointer stehende Rücksprungadresse herausgenommen (POP D) und statt dessen ein Sprungbefehl in das gewünschte Leitprogramm gegeben.

Programmteil „Pause"

Dieser Programmteil wird einmal als Unterprogramm aufgerufen und muß also mit einem RET-Befehl enden. Zum anderen wird das Programm durch die Pausencodes aus der Melodieliste

durch den bedingten Sprungbefehl angewählt. Hierbei muß am Ende des Programmteils durch einen Sprungbefehl zurückgesprungen werden. Die Pause selbst wird, wie man leicht sieht, durch eine 3fache Dekrementierschleife erzeugt. Es werden abgezählt: im Akkumulator der Tempokoeffizient, im Register E die decodierte Pausenlängenzahl und im Register D die Zahl FFH.

Die Teilprogramme für Spielanweisungen

Die Schaltung für die elektronische Umschaltung von Diskant auf Baß ist in Bild 1 dargestellt. Das Potential von D_4 des Ausgangskanals schaltet entweder D_0 (Diskant) oder D_2 (Baß) des gleichen Kanals auf den Ausgangsverstärker. In dem Umschaltprogramm, das von dem Code der Melodieliste aufgerufen wird, ist also das Ausgabe-Byte, welches im Tonerzeugungsprogramm für den Impulswechsel ausgegeben wird, vorzubereiten, indem 00 (Diskant) oder 10 (Baß) in eine RAM-Speicherstelle abgelegt wird.

Im Tonerzeugungsprogramm wird dieses Byte aus dem Speicher geholt und zur schnelleren Verfügbarkeit nun im Register L aufbewahrt. Der Impulswechsel erfolgt durch jedesmaliges Inkrementieren dieses Bytes. Um zu verhindern, daß nun der Überlauf beim 16. Inkrementierschritt das Umschaltbit zerstört, wird das Byte nach dem Inkrementieren noch mit 17H UNDiert. Dadurch bleiben D_0... D_2 und D_4 unverändert, aber D_3 wird immer wieder auf Null gesetzt und ein Überlauf durch dieses Bit damit verhindert.

Wiederholung-Anfang

Der augenblickliche Stand des Listenzeigers (Rp. HL) und die Zahl 02 auf einem weiteren Speicherplatz werden gespeichert.

Wiederholung-Ende

Dieser Code muß zweimal gesetzt werden. Sind beide Durchgänge der Wiederholung gleich, kommt an die Stelle des Wiederholungs-Ende-Zeichens zweimal 0B. Unterscheiden sich die Wiederholungen in der Endung, so sind die Code wie folgt zu setzen: Wiederholungsanfang = 0A, jeweils am Anfang und Ende des Teils, der beim ersten Durchgang anders gespielt werden soll als beim zweiten Durchgang, einmal 0B.

Im Programm wird beim ersten Durchgang die gespeicherte Zahl 02 geholt und dekrementiert. Das Ergebnis ist positiv, es wird also das Programm mit dem nächsten Listenwort fortgesetzt. Beim zweiten Erscheinen von 0B ist das Ergebnis des Dekrementierens Null. Dann wird HL auf den am Wiederholungsanfang gespeicherten Anfangswert gesetzt, und es beginnt die Wiederholung.

Beim Wiedererreichen des ersten 0B – jetzt der dritte Durchgang durch den Programmteil – ist das Ergebnis des Dekrementierens negativ, beide Sprungbedingungen sind also nicht erfüllt. In einer Schleife wird in der Melodieliste das nächste „0B" gesucht und dort in der Melodieliste fortgefahren.

Strophenende

Beim Erreichen von „Strophenende" wird der Listenzeiger auf den Anfangswert gesetzt und die Strophenzahl dekrementiert. Steht diese auf Null, ist das Programm zu Ende. Da der Computer sich hier noch in einem Unterprogramm befindet, wird die im Stackpointer stehende Rücksprungadresse gelöscht (POP D). Dann kann je nach der Organisation des Anwendergeräts ein HALT-Befehl oder ein Sprungbefehl kommen. Beim Verfasser bewirkt der RET-Befehl am Ende eines Anwenderprogramms automatisch die Rückkehr in das Monitorprogramm.

Schlußbemerkung

Die Programmierung des Ausgabekanals oder der Ausgabekanäle muß der Anwender nach seinen eigenen Erfordernissen vornehmen. Ein so umfangreiches Programm läuft natürlich selten auf Anhieb. Die Aufteilung in Teilprogramme erleichtert aber die Fehlersuche, weil man die verschiedenen Funktionen einzeln überprüfen kann. Zum Schluß sei noch als Beispiel die Melodieliste für das Volkslied „Horch, was kommt von draußen rein" angegeben. In ihr kommen sämtliche Spielanweisungen vor.

Zusätzliche Befehle beim 6502

Sollten Sie den Mikroprozessor 6502 von MOS Technology in Verwendung haben, so haben Sie ihn bisher vermutlich erheblich unterschätzt, was seinen Befehlsvorrat angeht. Wegen der Struktur der Befehlsdecodierung auf dem Chip werden manche Hexadezimalzahlen, die nicht im Programmierhandbuch enthalten sind, als u. U. recht nützliche Befehle interpretiert und ausgeführt. Sie stellen Bitmuster-Kombinationen „offizieller" Operationscodes dar und sind in der *Tabelle* aufgelistet. Nur eines noch: Manche Hexadezimal-Zahlen führen – als Befehl interpretiert – dazu, daß der Prozessor nur noch mit einem Reset-Befehl „aus dem Weltraum" zurückgeholt werden kann, wie z. B. 12 Befehle, deren letztes Halbbyte (niederwertiges „Nibble") 2 ist.

Übrigens wäre es interessant, herauszufinden, ob auch andere Mikroprozessoren mehr können, als ihre Hersteller wissen – eine lohnende Aufgabe für lange Winterabende. *Fe.*

Literatur

Gordon, H. T.: The XF and X7 Instructions of the MOS 6502. Byte, Dez. 1977.

Zusätzliche 6502-Operationscodes

Hex-Code	*Wirkung*
A7 aa	Akku und X-Register werden mit dem Inhalt der Zero-Page-Adresse aa geladen
87 aa	Das Ergebnis einer UND-Funktion zwischen Akku und X-Register wird an die Zero-Page-Adresse aa gespeichert
97 aa	wie 87, jedoch wird das UND-Ergebnis an die Zero-Page-Adresse aa + Y gespeichert
9E aaaa	Das Ergebnis einer UND-Funktion zwischen dem X-Register und der Konstante 02 wird an die Adresse aaaa gespeichert
9F aaaa	Das Ergebnis einer UND-Funktion zwischen dem Akku, dem X-Register und der Konstante 02 wird an die Adresse aaaa gespeichert
E7 aa	Der Inhalt der Zero-Page-Adresse aa wird um eins erhöht; das Resultat wird dann vom Akkuinhalt subtrahiert
C7 aa	Der Inhalt der Zero-Page-Adresse aa wird um eins erniedrigt und anschließend mit dem Akkuinhalt verglichen; die Status-Flags werden wie bei CMP-Befehlen beeinflußt
AB dd	Akku und X-Register werden mit den Daten dd geladen

Eckart Märkel

EPROM-Programmiergerät

2. Teil

Zur Formung des Programmierimpulses wird der in Bild 4 rechts dargestellte einfache Programmierimpulsgenerator verwendet, der die in Bild 3 vorgegebene Impulsform annähern soll. Für die Impulsform zeitbestimmend sind die Widerstände 1,5 kΩ und 2,2 kΩ sowie der 330-pF-Kondensator. An die Transistoren werden keine besonderen Anforderungen gestellt; es ist lediglich auf eine ausreichende zulässige Kollektor-Emitterspannung zu achten. Der PNP-Transistor leitet den bei niedrigem Pegel maximal 3 mA betragenden Ausgangsstrom aus dem Programmiereingang ab, um eine unter 1 V liegende Eingangsspannung zu erreichen. Die Spannung für den hohen Pegel wird mit der Zenerdiode stabilisiert, evtl. reihengeschaltete Dioden dienen zur Feineinstellung.

Die Programmierung mit diesem Gerät dauert länger als die vom Hersteller als Richtwert angegebene Dauer von 100 Sekunden. Das liegt daran, daß der Programmierimpuls nur während eines Viertels der gesamten Programmierzeit gegeben wird, obwohl ein größeres Tastverhältnis möglich wäre. Für die gedachte Anwendung ist das aber belanglos.

Noch ein Hinweis zum Einsatz des Gerätes mit Mikrocomputern: Möchte man einen Speicherinhalt aus einem Mikrocomputer programmieren, so geht das am einfachsten mit direktem Speicherzugriff (DMA = direct memory access), den die meisten Mikrocomputer erlauben. Ein besonderes Signal bringt Daten- und Adreßausgänge in den hochohmigen Zustand, so daß das Programmiergerät an diese Busse angeschlossen werden kann. Man muß nur noch für die Selektion des richtigen Speichers sorgen.

Löschen des EPROM

Zum Schluß noch einige Hinweise zum Löschen: Nach Herstellerangaben ist zum Löschen eine UV-Dosis von 15 Ws/cm² bei einer Wellenlänge von 2537 Å nötig. Bei üblichen UV-Lampen bedeutet das 10...45 min Bestrahlungszeit bei 3 cm Abstand. Es muß für eine ausreichende Löschung gesorgt werden, da sonst – besonders bei höherer Arbeitstemperatur – sich das alte Bitmuster dem neuen überlagern könnte.

Der Erfolg einer Löschung kann mit diesem Programmiergerät auf einfache Weise überprüft werden: Man führt die oben beschriebene Prüfoperation durch, jedoch ohne einen externen Speicher angeschlossen zu haben. Wegen der TTL-Eigenschaft liegen die Dateneingänge dann auf „1", das bedeutet, daß überprüft wird, ob auch überall im EPROM Einsen stehen –, ein Kennzeichen für ordnungsgemäße Löschung.

Das Gerät wurde mit der in der Digitaltechnik bewährten Fädeltechnik aufgebaut. Integrierte Schaltkreise, Fassungen, Steckleisten und ähnliches lassen sich damit recht gut verbinden, diskrete Bauteile allerdings weniger; letztere kann man auf einer Platine üblicher Bauart getrennt unterbringen (Bedienungselemente und Programmierimpulsgenerator). Günstig ist, daß man recht leicht nachträglich Korrekturen vornehmen kann. Verbesserungswürdig wäre vielleicht noch der Lötlack dieses Drahtes: Er schmilzt schwer weg, so daß längere Lötzeiten nötig sind und gelegentlich „kalte" Lötstellen entstehen. *Tabelle 2* faßt noch einmal die wichtigsten Daten des Gerätes zusammen, *Tabelle 3* nennt die benötigten TTL-Schaltkreise.

Tabelle 3: Verwendete TTL-Schaltkreise

Stück	Typ	Funktion
1	7400	4 NAND-Gatter mit je 2 Eingängen
1	7404	6 Inverter
6	7405	6 Inverter mit offenem Kollektor
1	7420	2 NAND-Gatter mit je 4 Eingängen
2	7430	1 NAND-Gatter mit 8 Eingängen
1	7473	2 JK-Flipflops
1	7474	2 D-Flipflops
2	7486	4 Exklusiv-ODER-Gatter mit je 2 Eingängen
6	7493	4-bit-Binärzähler
2	74125	4 Tristate-Gatter
1	74132	4 NAND-Schmitttrigger mit je 2 Eingängen

Literatur

[1] Intel Data Catalog

[2] Siemens-Datenbuch: Mikroprozessor-Bausteine, System SAB 8080

Tabelle 2: Wichtigste Daten des Gerätes

Stromversorgung

–5 V ±5%	45 mA	(nur für EPROM)
+5 V ±5%	600 mA	
+12 V ±5%	65 mA	(nur für EPROM)
+32 V	25 mA	(Programmierspannung)

Adreßausgang	TTL-fan-out	7
Dateneingang	TTL-fan-in	2

Programmierdauer
ca. 7 min für 2708 (2704 die Hälfte)

Ein Bit oder 1 bit?

Bei digitaltechnischen Einheiten stiftet manchmal Klein- und Großschreibung Verwirrung. Kein Wunder – denn 1 kByte ist etwas anderes als 1 KByte, und auch ein Bit ist etwas anderes als 1 bit:

1 kByte = 1000 Byte, denn „k" bedeutet 1000. Allerdings: 1 kByte kommt in der Praxis kaum vor; eher schon **1 KByte** = 1024 Byte. Ein großes K bedeutet hier nämlich 1024, also 2^{10}.

1 bit ist die Einheit für eine digitale Ja-Nein-Aussage oder Speicherzelle; z. B. kann ein Byte (ein Datenwort) aus 8 bit bestehen. Aber:

Ein Bit stellt – abseits mathematischer Zusammenhänge – genau eine Ja-Nein-Entscheidung dar.

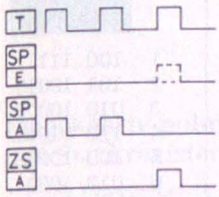
Hans Werner

Mikrocomputer

Eine kleine Einführung

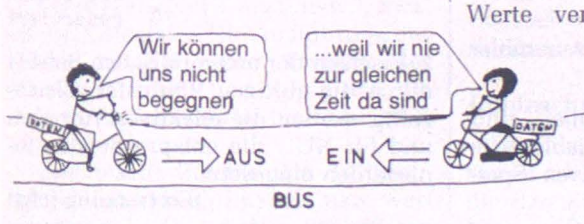
9.Teil

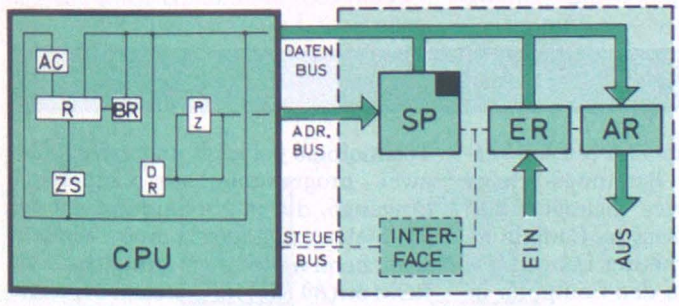
Bidirektionaler Bus



Ein- und Ausgabebus sind nie zur gleichen Zeit belegt. Beide können zu einem einzigen Bus vereint werden, auf dem die Daten in zwei Richtungen transportiert werden:

1. vom Speicher oder Eingangsregister in die CPU;
2. aus der CPU in den Speicher oder in das Ausgangsregister.

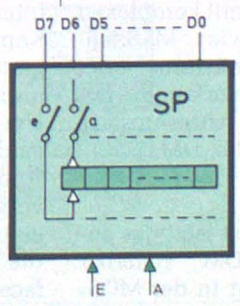




Die gleiche Änderung ist am Speicher zu bemerken. Alle Eingänge sind über die Schalter e (nur einer gezeichnet), alle Ausgänge über die Schalter a auf die Anschlüsse D0...D7 gelegt.

E	A	e	a	Funktion
1	0	1	0	Eingabe
0	1	0	1	Ausgabe
0	0	0	0	hochohmig

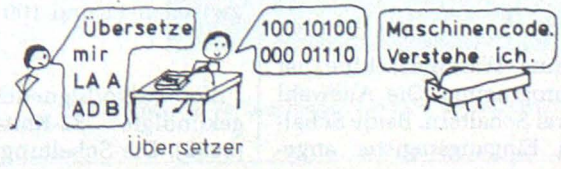
Schalter geschlossen = 1



Die Steuereingänge E, A behalten ihre bisherige Funktion, d. h., Steuerbus und Interface-Logik bleiben unverändert.

Insgesamt werden acht Anschlüsse gespart. Das ist besonders für die Hersteller von integrierten Schaltkreisen wichtig, denn jeder Anschluß vergrößert das Gehäuse. Heute ist fast ausschließlich der bidirektionale Datenbus (Zweiweg-Bus) anzutreffen.

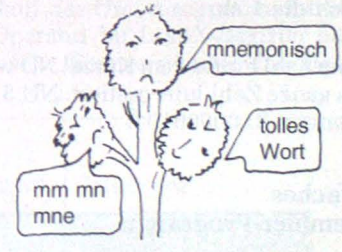
Jeder Computer kann nur binäre Werte verarbeiten. Damit die Programme übersichtlich und leicht zu lesen sind, wurden bisher die binären Darstellungen vermieden und leicht merkbare Kurzzeichen eingeführt. Diese sogenannten **Mnemonics** (Mneme = griechisch, Gedächtnis) müssen wieder in den Maschinencode umgesetzt werden (000 für AD usw.). Das kann mit Hilfe einer Tabelle oder durch einen Übersetzungsautomaten geschehen. Als Übersetzer kommt wieder ein Computer in Frage, der mit einem speziellen Programm die Mnemonics in den Maschinencode umsetzt.



Im einfachsten Fall wird mit einem Lochstreifenstanzer (Typenfeld wie Schreibmaschine) ein Lochstreifen angefertigt, der von einem Computer eingelesen und übersetzt wird. Das Maschinenprogramm ist danach als neuer Lochstreifen vorhanden, dessen Daten mit besonderen Geräten in den Speicher des Mikrocomputers eingeschrieben werden.

Assembler

Damit das Übersetzungsprogramm richtig arbeiten kann, müssen bestimmte Regeln und Vorschriften eingehalten werden. Sie werden vom Hersteller in einem Software-Handbuch herausgegeben. Die symbolische Schreibweise ist eng an den Maschinencode angelehnt, d. h. jeder Maschinenbefehl wird lediglich durch eine einfache Bezeichnung (Mnemonic) ersetzt.



Als weitere Erleichterung darf der Speicher mit Phantasienamen adressiert werden. Das Übersetzungsprogramm sucht sich selbständig die absoluten Adressen heraus. Solche Programmsprachen werden ASSEMBLER-SPRACHE genannt. Das dazugehörige Übersetzungsprogramm hat den gleichen Namen.

Assembler-Sprache:
maschinenorientierte Programmsprache.

Der Maschinencode wird durch mnemonische Bezeichnungen ersetzt. Adressen und Variablennamen dürfen symbolisch sein.

Assembler:
Programm zum Übersetzen der Assembler-Sprache in ein ablauffähiges Objektprogramm (Maschinencode). Ein Assemblerbefehl erzeugt einen Maschinenbefehl.

Mini-Software-Handbuch

- Jedes Programm muß mit dem Befehl AB beginnen. AB ist nur an den Assembler gerichtet und legt die Anfangsadresse des Objektprogramms fest (Maschinencode).
- Symbolische Adressen und Namen dürfen maximal 4 Zeichen haben. Verboten sind alle Kombinationen, die mit AD, SU...ST, AB und NU beginnen.
- Die Adresse eines Sprungbefehls muß sich direkt auf das Ziel beziehen. Der Assembler zieht automatisch die 1 ab.
- Eine Zahl hinter dem Kürzel NU wird als ganze Zahl interpretiert. NU 5 bedeutet z.B. 00000101.

Einfaches Assembler-Programm

Problem: Eine Waschmaschine hat zwei Waschprogramme. Die Auswahl erfolgt mit zwei Schaltern. Beide Schalter sind am Eingangsregister angeschlossen (D0 und D1). Ist kein Schalter gesetzt, soll eine Warteschleife ausgeführt werden. Die eigentlichen Waschprogramme bestehen aus jeweils 3 Befehlen. Sie sind nur durch Punkte angedeutet.

Adresse	Befehl	Erklärung
	AB 1	Das Programm soll mit 1 beginnen
ANF	LA 31	Schalterstellungen in den AC laden
	SP N	in Zelle N speichern
	UN EINS	Ersten Schalter selektieren
	JP PR 1	Wenn (AC)=1,nach Progr.1 springen
	LA N	Schalterstellungen laden (aus N)
	UN ZWEI	Zweiten Schalter selektieren
	JP PR 2	Wenn (AC)=1,nach Progr. 2 springen
	LA EINS	damit (AC) ungleich 0
	JP ANF	zum Programmanfang ANF springen
PR 1	.. ....	
	.. ....	Programm 1
	.. ....	
PR 2	.. ....	
	.. ....	Programm 2
	.. ....	
EINS	NU 1	Inhalt von Zelle EINS = 1=00000001
ZWEI	NU 2	Inhalt ZWEI = 2=00000010

Der symbolischen Adresse ANF wird die effektive Adresse 1, der Adresse N der Platz 18 am Programmende reserviert usw.; Sprungadressen werden automatisch um 1 geringer gewählt (Befehl 9).

Adressen	
symb.	effektiv
ANF	1
N	18
EINS	16
PR 1	10
ZWEI	17
PR 2	13

	MASCHINEN-PROGRAMM
1	100 11111
2	101 10010
3	010 10000
4	110 01010
5	100 10010
6	010 10001
7	110 01101
8	100 10000
9	110 00000
10	
11	Progr. 1
12	
13	
14	Progr. 2
15	
16	000 00001
17	000 00010
18	reserviert

Übersetzen in das Objektprogramm

Grundsätzlich führt der Assembler zwei Schritte aus:

1. Alle symbolischen Namen und Adressen werden in eine Variablenliste eingetragen und die effektiven Speicheradressen ermittelt.

2. Ersetzen der mnemonischen Befehle durch die gültigen Bitmuster. Gleichzeitig werden die effektiven Adressen und für NU... die entsprechenden Binärzahlen eingesetzt.

(Fortsetzung folgt)

µC-Nachrichten

Die Herausgeber des im Müller-Verlag erscheinenden Beratungs-Briefes „Verkaufsleiter-Service" befragten 207 Firmen über den Einsatz des **Computers in Verkauf und Marketing.** Danach setzen 86 % der Firmen den Computer in diesen beiden Bereichen ein; rund zwei Drittel verwenden ihn zum Erstellen von Umsatz-Statistiken. Allerdings beschäftigen nur 11 der 207 befragten Firmen ihren Computer mit komplexen Aufgabenstellungen, wie Marktanteils-Berechnungen, Preistests oder Kontrolle der Werbewirkung. Die Mehrzahl der Firmen hat Jahresumsätze zwischen 11 und 100 Mio. DM.

□

Von Valvo/Signetics ist jetzt das angekündigte **32-Kbit-ROM** lieferbar (2632). Die Schaltung ist in der MOS-Technologie gefertigt und verfügt über zwei programmierbare Chip-Select-Eingänge, die in Verbindung mit den Tri-State-Ausgängen eine einfache Speichererweiterung gestatten. Ein ROM mit 64 Kbit im gleichen 24poligen Gehäuse ist bereits in Vorbereitung.

□

Für den **TRS-80** von Tandy/Radio Shack ist in den USA jetzt ein Telefon-Interface (Modem mit akustischem Koppler) lieferbar, mit dem sich Programme und Datenblöcke über das normale Telefonnetz übertragen lassen. Um ein solches Gerät auch in Deutschland einsetzen zu können, müssen noch Änderungen an der Übertragungsart vorgenommen werden; außerdem ist eine FTZ-Genehmigung erforderlich, die Tandy für 1979 erwartet. Das Interface soll etwa 300 DM kosten.

Falk Domzig

Boolesche Algebra mit Taschenrechnern

Im folgenden Beitrag soll gezeigt werden, wie ein programmierbarer Taschenrechner mit algebraischen Befehlen boolesche Verknüpfungen durchführen kann. Ein Rechner mit der Eingabe in Form der umgekehrten polnischen Notation bietet dabei eine besonders einfache Programmierung.

Umgekehrte polnische Notation und Stackregister

Zum Verständnis der Programme für boolesche Verknüpfungen sei kurz die Arbeitsweise eines Rechners mit umgekehrter polnischer Notation (UPN) und Stackregister erläutert. Ein solcher Rechner, z. B. HP-19, HP-29 oder HP-67, hat entsprechend *Bild 1* fünf Arbeitsspeicher: X, Y, Z, T und Last-X-Register. Angezeigt wird stets das X-Register. Jeder Befehl, der nur einen Operand verändert (wie z. B. sin x, log x, usw.), wirkt auf das X-Register. Die Ausführung des Befehls bewirkt einen Transfer des Operanden in das Last-X-Register, die anderen Registerinhalte bleiben unverändert.

Algebraische Befehle wie Addition. Subtraktion, Multiplikation usw. verknüpfen zwei Operanden. Sie müssen dazu in die Register Y und X eingegeben werden. Dieses Laden erfolgt in der Weise, daß zuerst der Operand Y eingetastet wird (zunächst Anzeige aus dem X-Register), der „Enter"-Befehl bewirkt ein Duplizieren in das Y-Register, durch die jetzt folgende Eingabe des zweiten Operanden wird das X-Register überschrieben. Der nun folgende Operationsbefehl verarbeitet beide Operanden. Das Ergebnis der Rechnung erscheint im X-Register. Gespeichert wird dabei der X-Operand im Last-X-Register. Die Inhalte der Register Z und T rutschen bei Ausführung des Operationsbefehls nach Y und Z vor (das T-Register wird dabei nicht gelöscht).

Boolesche Funktionen

Die hier behandelten Zustandsfunktionen arbeiten mit binären Elementen, die den Zustand „1" oder „0" haben können. *Bild 2* enthält alle möglichen Verknüpfungen zweier verschiedener Elemente. Abgesehen von den Konstanten lassen sich durch die Funktionen Negation, Konjunktion (Und) und Disjunktion (Oder) alle anderen Verknüpfungen darstellen.

Programme boolescher Funktion

Die nun dargestellten Unterprogramme für die entsprechenden booleschen Funktionen wirken analog der Arbeitsweise eines Rechners mit UPN nur auf das X-Register oder bei Befehlen mit zwei Operanden auf die Register X und Y. Eventuelle Zwischenergebnisse, die das Stack-Register speichert, müssen erhalten bleiben. Eine Zwischenspeicherung des Stackinhalts vor Ausführung des Unterprogrammes und anschließender Rückspeicherung wird wegen des erhöhten Programmieraufwandes nicht durchgeführt. Nach dem Ablauf des Unterprogrammes muß der X-Operand im Last-X-Register enthalten sein. Ebenso wie bei den algebrai-

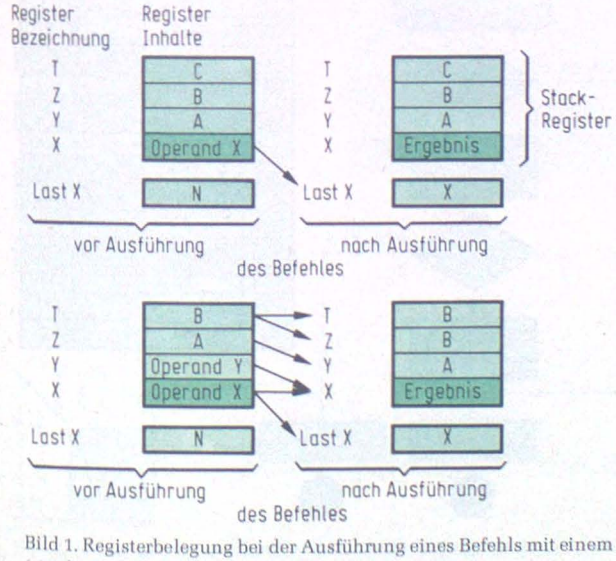


Bild 1. Registerbelegung bei der Ausführung eines Befehls mit einem (a) oder zwei Operanden (b)

X 0011 / Y 0101	Boolescher Ausdruck	Bezeichnung
F_0 0000	0	Konstante 0
F_1 0001	$X \vee Y$	Konjunktion (Und)
F_2 0010	$X \wedge \overline{Y}$	
F_3 0011	X	Konstante X
F_4 0100	$\overline{X} \wedge Y$	
F_5 0101	Y	Konstante Y
F_6 0110	$(\overline{X} \vee \overline{Y}) \wedge (X \vee Y)$	Antivalenz (Exklusiv-Oder)
F_7 0111	$X \vee Y$	Disjunktion (Oder)
F_8 1000	$\overline{X \vee Y}$	Weder - noch (Nor)
F_9 1001	$(\overline{X} \wedge \overline{Y}) \vee (X \wedge Y)$	Äquivalenz
F_{10} 1010	$\overline{Y}$	Negation
F_{11} 1011	$X \vee \overline{Y}$	Implikation
F_{12} 1100	$\overline{X}$	Negation
F_{13} 1101	$\overline{X} \vee Y$	Implikation
F_{14} 1110	$\overline{X \wedge Y}$	Sheffer - Strich (Nand)
F_{15} 1111	1	Konstante 1

Bild 2. Mögliche Verknüpfungen von zwei Zustandsgrößen

schen Berechnungen können mit dem Last-X-Operanden Kettenrechnungen ausgeführt werden.

Die Programme sind durch ein allgemeines Flußdiagramm und eine Programmliste dargestellt, die für den Hewlett-Packard-Rechner HP 67/97 Anwendung findet. Die Programme lassen sich für andere Rechner mit UPN leicht übernehmen.

Negation

Bei Ausführung einer Negation darf nur das X-Register geändert werden. Es soll zu „1" werden, wenn es eine Null enthält und umgekehrt. Das Programm besteht im wesentlichen aus einer Abfrage (bedingte Programmverzweigung): Ist das X-Register gleich Null? Ist das Register ungleich Null, so wird es mit dem Befehl Clear X (ClX) gelöscht. Damit wird es zu Null. Ist das „Register 1", so wird es mit ClX gelöscht und anschließend eine „1" eingegeben. Die nach dem Löschbefehl folgende Eingabe bewirkt keine Veränderung der restlichen Stack-Register (also kein „Stacklift").

Bei den bisher beschriebenen Programmbefehlen wird keine Operation ausgeführt, welche einen Transfer des X-Registers ins Last-X bewirkt. Mit dem Befehl X^2 kann diese Überschreibung vorgenommen werden. Da die Elemente einer Funktion nur den Wert „0" oder „1" haben können, verändert sich das X-Register durch das Quadrieren nicht. Der Befehl X^2 wird dem Programm vorangestellt (*Bild 3*).

Konjunktion (Und)

Keinerlei Probleme ergeben sich bei der Und-Verknüpfung. Der algebraische Befehl der Multiplikation wirkt in gleicher Weise: $0 \times 0 = 0$, $0 \times 1 = 0$, $1 \times 0 = 0$, $1 \times 1 = 1$ (*Bild 4*).

Disjunktion (Oder)

Eine disjunktive Verknüpfung kann durch algebraische Addition vorgenommen werden. Ausgenommen ist der Fall, daß beide Operanden gleich „1" sind. Bei diesem Zustand erbringt nur der Multiplikationsbefehl das richtige Ergebnis. Das gesamte Flußdiagramm ist in *Bild 5* aufgezeichnet. Der Austauschbefehl zwischen den Registern X und Y nach der zweiten Programmverzweigung (ist X ungleich Null?) ist notwendig für den richtigen Transfer des X-Operanden in das Last-X-Register.

Nand und Nor

Für Schaltungs-Simulationen werden des öfteren die Verknüpfungen Nand und Nor benötigt. Obwohl sie durch die drei zuvor aufgezeigten Programme berechnet werden können, seien ihre Flußdiagramme mit Befehlslisten angegeben. Bei Verwendung dieser Unterprogramme (*Bild 6*) enthält das Last-X-Register nach Ausführung des Programmes den richtigen Wert des X-Operanden. Werden die Programme Und und Negation für das Nand bzw. Oder und Negation für das Nor aneinandergereiht, so enthält das Last-X-Register den X-Operanden des letzten Unterprogrammes.

Exklusiv-Oder

Durch den Subtraktionsbefehl kann die Exklusiv-Oder-Funktion (auch Antivalenz) nachgebildet werden. Ist das Y-Register 0, das X-Register 1, so wird das Ergebnis bei Subtraktion -1. Das Exklusiv-Oder muß aber bei dem geschilderten Zustand eine 1 erbringen. Durch die Abfrage im Programm nach negativem Ergebnis kann dieser Fall erkannt werden. Der Befehl Change Sign (CHS) wirkt auf das X-Register, ohne dabei das Last-X-Register zu verändern (nicht so beim Befehl: Betrag von x – ABS); siehe dazu das Flußdiagramm in *Bild 7*.

Berechnung von komplexen booleschen Funktionen

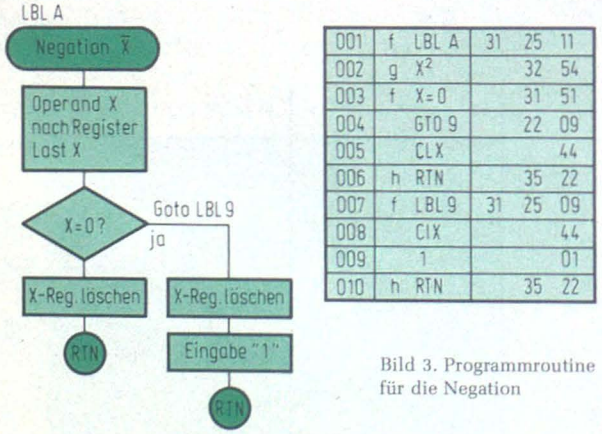
Nachstehend soll gezeigt werden, wie ein Programm einer booleschen Funktion mit Hilfe der besprochenen Unterprogramme aufgestellt werden kann.

Das Beispiel soll sein:

$Q = f_{(A,B,C,D)} = (A \wedge B \wedge \overline{C} \wedge D) \vee (\underline{A} \wedge B \wedge C \wedge D) \vee (\overline{A} \wedge B \wedge C \wedge D) \vee (\overline{A} \wedge B \wedge C \wedge \overline{D})$

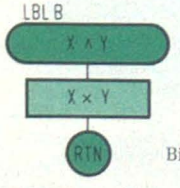
Die Karnaugh-Tabelle in *Bild 8* zeigt für alle Zustände der Veränderlichen die Ergebnisse. Ordnen wir die Speicherplätze 4...1 den Veränderlichen A...D zu, so läßt sich folgendes Programm aufstellen (*Bild 9*).

Mit den Unterprogrammen kann die Programmierung also direkt entspre-



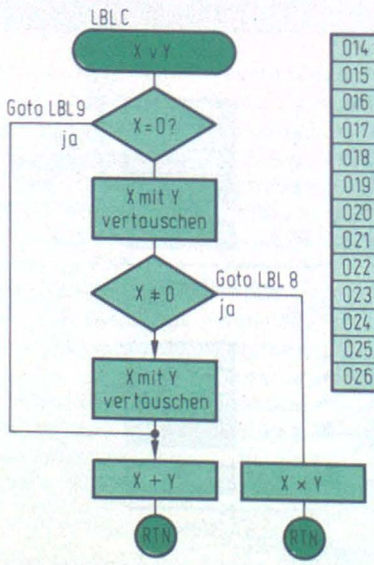
001	f	LBL A	31 25	11
002	g	X^2	32	54
003	f	X=0	31	51
004		GTO 9	22	09
005		CLX		44
006	h	RTN	35	22
007	f	LBL 9	31 25	09
008		CLX		44
009		1		01
010	h	RTN	35	22

Bild 3. Programmroutine für die Negation



011	f	LBL B	31 25	12
012		×	71	
013	h	RTN	35	22

Bild 4. Die UND-Funktion als Programm



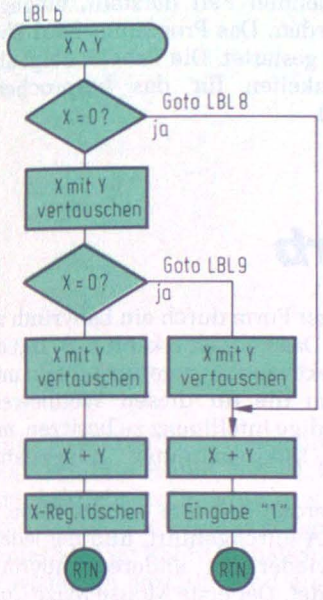
014	f	LBL C	31 25	13
015	f	X=0	31	51
016		GTO 9	22	09
017	h	X↔Y	35	52
018	f	X≠0	31	61
019		GTO 8	22	08
020	h	X↔Y	35	52
021	f	LBL 9	31 25	09
022		+		61
023	h	RTN	35	22
024	f	LBL 8	31 25	08
025		×		71
026	h	RTN	35	22

Bild 5. Unterprogramm für die Oder-Funktion

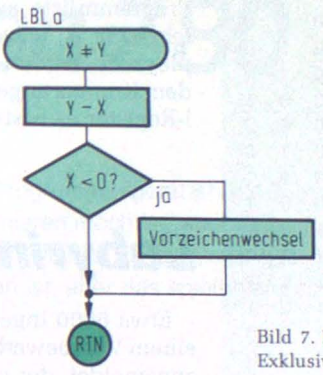
chend der Funktion erfolgen. Die Speicher der Veränderlichen müssen vor der Berechnung mit ihren Ausgangswerten geladen werden.

Eine Programmroutine, in welche das boolesche Programm für $Q = f(F_1 ... F_9)$ eingekleidet ist, soll das Laden der Speicher erleichtern. Das Anzeigeformat des Rechners begrenzt diese Routine auf die Berechnung von Funktionen bis zu 9 Veränderlichen.

Durch das nun folgende Programm soll eine Eingabe der Veränderlichen in Form einer mehrstelligen Dualzahl ermöglicht werden. Das Programm lädt die niederste Stelle der Dualzahl in den Speicher 1, die folgende Stelle in den Speicher 2 usw. (*Bild 10*). Mit dem unbedingten Sprungbefehl GTO 1 springt die Routine in das Programm der booleschen Funktion $Q = f(F_i)$, die mit dem

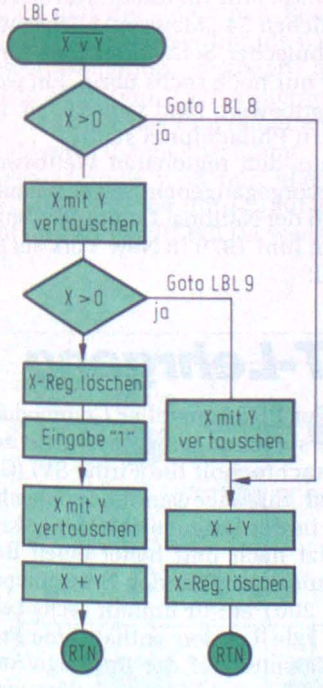


027	g	LBL b	32	25	12
028	f	X=0		31	51
029		GTO 8		22	08
030	h	X↔Y		35	52
031	f	X=0		31	51
032		GTO 9		22	09
033	h	X↔Y		35	52
034		+			61
035		CLX			44
036	h	RTN		35	22
037	f	LBL 9	31	25	09
038	h	X↔Y		35	52
039	f	LBL 8	31	25	08
040		+			61
041		CLX			44
042		1			01
043	h	RTN		35	22



061	g	LBL a	32	25	11
062		–			51
063	f	X<0		31	71
064		CHS			42
065	h	RTN		35	22

Bild 7. Programm zur Simulation eines Exklusiv-Oder-Gatters



044	g	LBL c	32	25	13
045	f	X>0		31	81
046		GTO 8		22	08
047	h	X↔Y		35	52
048	f	X>0		31	81
049		GTO 9		22	09
050		CLX			44
051		1			01
052	h	X↔Y		35	52
053		+			61
054	h	RTN		35	22
055	f	LBL 9	31	25	09
056	h	X↔Y		35	52
057	f	LBL 8	31	25	08
058		+			61
059		CLX			44
060	h	RTN		35	22

◀ Bild 6. Die Nand- und Nor-Funktionen

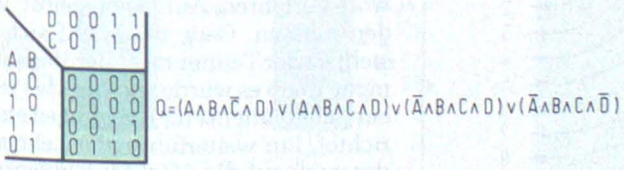


Bild 8. Karnaugh-Tabelle für eine komplexe boolesche Funktion

091	f	LBL 1	31	25	01	
092		RCL 4		34	04	Variable A
093		RCL 3		34	03	B
094	f	GSB B	31	22	12	Und
095		RCL 2		34	02	C
096	f	GSB A	31	22	11	Negation C̄
097	f	GSB B	31	22	12	Und
098		RCL 1		34	01	D
099	f	GSB B	31	22	12	Und
100		RCL 4		34	04	A
101		RCL 3		34	03	B
102	f	GSB B	31	22	12	Und
103		RCL 2		34	02	C
104	f	GSB B	31	22	12	Und
105		RCL 1		34	01	D
106	f	GSB B	31	22	12	Und
107	f	GSB C	31	22	13	Oder
108		RCL 4		34	04	A
109	f	GSB A	31	22	11	Negation Ā
110		RCL 3		34	03	B
111	f	GSB B	31	22	12	Und
112		RCL 2		34	02	C
113	f	GSB B	31	22	12	Und
114		RCL 1		34	01	D
115	f	GSB B	31	22	12	Und
116	f	GSB C	31	22	13	Oder
117		RCL 4		34	04	A
118	f	GSB A	31	22	11	Negation Ā
119		RCL 3		34	03	B
120	f	GSB B	31	22	12	Und
121		RCL 2		34	02	C
122	f	GSB B	31	22	12	Und
123		RCL 1		34	01	D
124	f	GSB A	31	22	11	Negation D̄
125	f	GSB B	31	22	12	Und
126	f	GSB C	31	22	13	Oder
127		GTO 2		22	02	

Bild 9. ▶ Programm zu der Funktion in Bild 8

Befehl GTO 2 endet. Das Programm unter der Marke LBL 2 ist eine Ausgaberoutine, deren Folge es ist das Ergebnis in Form $Q, F_9F_8F_7F_6F_5F_4F_3F_2F_1$ anzuzeigen. Die Veränderlichen stehen also im Nachkommateil; die dem höchsten Speicher zugeordnete Variable steht zuerst (mathematisch dargestellt: $Q\ 10^0 + F_S\ 10^{-1} + F_{s-1}\ 10^{-2} + F_{S-2}\ 10^{-3} + ...$, wobei S die Anzahl der Veränderlichen ist.)

In *Bild 11* ist das Flußdiagramm mit Programmliste aufgezeichnet. Der Befehl DSP (i) steuert die Anzeige (Display). Es werden so viele Stellen nach dem Komma angezeigt, wie die Zahl im I-Register es bestimmt. Der Befehl INT (integer) beseitigt den Nachkommateil der Zahl. Der Befehl DSZ I (decrement) erniedrigt das I-Register um 1, und RTN (return) ist der Schlußbefehl. Bevor die Funktion berechnet werden soll, muß in den Speichern E die Anzahl der Veränderlichen S eingegeben werden. Danach kann die Dualzahl N, die die Zustände der Veränderlichen für den zu berechnenden Fall darstellt, eingegeben werden. Das Programm wird über Label E gestartet. Die *Tabelle* zeigt alle Möglichkeiten für das besprochene Beispiel.

Ergebnisse für die Funktion

$Q = (A \wedge B \wedge \overline{C} \wedge D) \vee (A \wedge \overline{B} \wedge C \wedge D) \vee (\overline{A} \wedge B \wedge C \wedge D) \vee (\overline{A} \wedge B \wedge C \wedge \overline{D})$

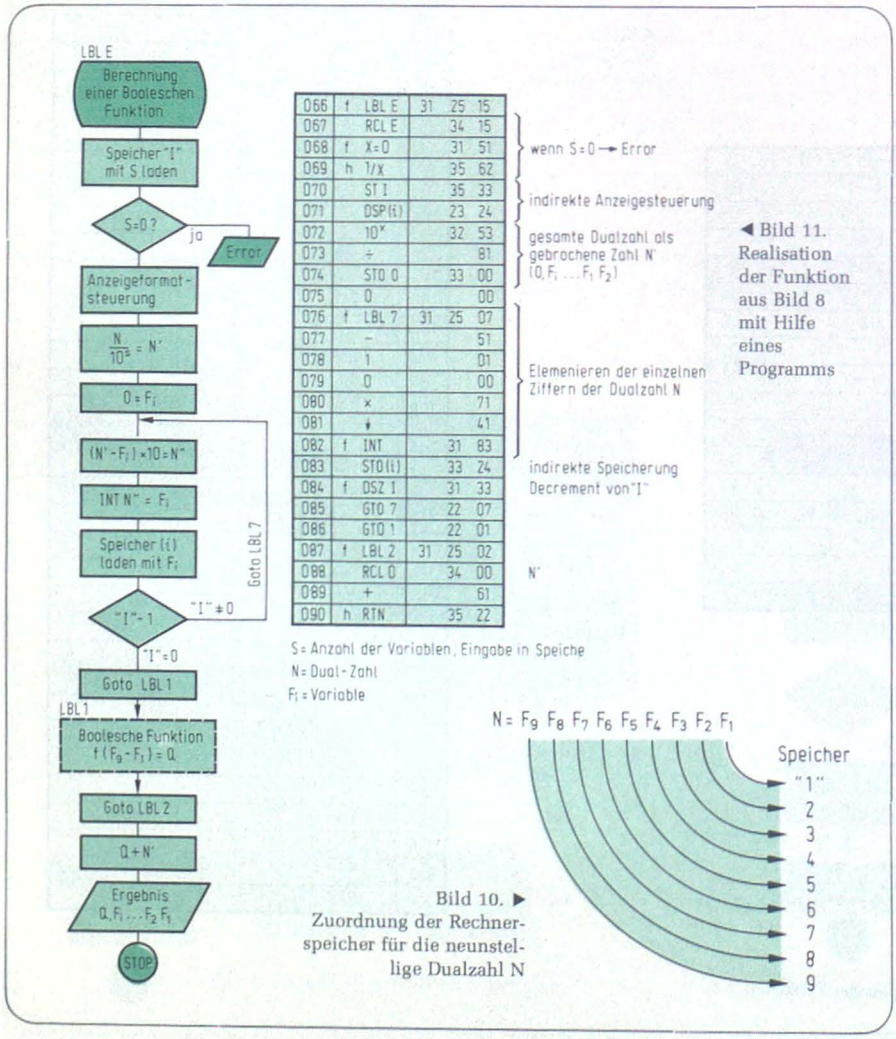
Eingabe				Ausgabe				
				0.	0	0	0	0
0	0	0	0	0.	0	0	1	0
0	0	1	0	0.	0	0	1	1
0	0	1	1	0.	0	0	0	1
0	0	0	1	0.	1	0	0	0
1	0	0	0	0.	1	0	1	0
1	0	1	0	0.	1	0	1	1
1	0	1	1	0.	1	0	0	1
1	0	0	1	0.	1	1	0	0
1	1	0	0	0.	1	1	1	0
1	1	1	0	1.	1	1	1	1
1	1	1	1	1.	1	1	0	1
1	1	0	1	0.	0	1	0	0
0	1	0	0	1.	0	1	1	0
0	1	1	0	1.	0	1	1	1
0	1	1	1	0.	0	1	0	1
0	1	0	1					
A	B	C	D	Q,	A	B	C	D
Veränderliche				Q = Ergebnis				

Labyrinth-Wettbewerb

Etwa 6000 Ingenieure haben sich zu einem Wettbewerb ganz besonderer Art angemeldet, der vom IEEE (Institute of Electrical And Electronics Engineers) ausgeschrieben ist. Er nennt sich „Electronic Mouse Maze Contest" und stellt die Aufgabe, mit einem Bodenfahrzeug beliebiger Form durch ein Labyrinth zu finden. Das Fahrzeug kann z. B. mit einem Elektromotor angetrieben sein und darf, um die für diesen Wettbewerb notwendige Intelligenz zu besitzen, mit einem Mikrocomputer ausgestattet sein.

Es werden mehrere Wettbewerbe in den USA durchgeführt, und bei jedem wird wieder ein anderes Labyrinth verwendet. Der erste Mouse Maze Contest fand in Anaheim in Kalifornien vom 6. bis 8. Juni 1978 statt; von den ursprünglichen 54 „Mäusen" blieben wegen technischer Schwierigkeiten ihrer Erbauer nur noch sechs übrig. Ein weiterer Wettbewerb fand vom 25. bis 26. August in Philadelphia statt.

Die aus den regionalen Wettbewerben hervorgegangenen Sieger nehmen während der National Computer Conference im Juni 1979 in New York am Finale teil.



◀ Bild 11. Realisation der Funktion aus Bild 8 mit Hilfe eines Programms

Bild 10. ▶ Zuordnung der Rechnerspeicher für die neunstellige Dualzahl N

PET-Lehrgang

Was der PET-Hersteller Commodore trotz Versprechungen bisher nicht zustandebrachte, holt die Firma SVI (Gesellschaft für angewandte Problemlösungen in der Datenverarbeitung, Krefeld) jetzt nach und bietet einen BASIC-Fernlehrgang für den Heimcomputer PET 2001 an. Er umfaßt sechs Lektionen; jede Lektion enthält eine Programmkassette, auf der im Frage/Antwort-Verfahren Aufgaben gelöst werden müssen. Ganz allein auf sich gestellt ist der Teilnehmer allerdings auch nicht, denn es wurde werktags bis 19.00 ein Telefondienst für Rückfragen eingerichtet. Ein weiterführender Lehrgang, der auch auf die 6502-Maschinensprache und auf Interface-Probleme eingeht, ist bereits in Vorbereitung.

Jörg Ullrich

REPROM-Programmiergerät für den Z 80

Das hier beschriebene Programmiergerät ist eine Billigversion für alle, die gelegentlich ein REPROM programmieren möchten, aber keine 400 DM (oder mehr) für ein Programmiergerät ausgeben wollen. Es sei gleich gesagt, daß dieses Gerät nicht sehr komfortabel zu bedienen ist, aber das spielt hier auch keine große Rolle.

Die Z80-PIO enthält zwei Schnittstellen à 8 bit (Port A und Port B). In dem hier beschriebenen Programm wird aus Port A der PIO die Adresse ausgegeben, die die Daten im REPROM haben sollen, die wiederum anschließend an Port B ausgegeben werden.

Ist die Ausgabe der Daten aus Port B erfolgt, gibt die PIO einen Ready-Impuls an IS1 ab. Dieser Impuls wird von diesem Monoflop verzögert an das zweite Monoflop (IS2) weitergegeben, das die Dauer des Programmierimpulses (hier ca. 0,7 ms) bestimmt.

Dieser Impuls wird nun über einen Verstärker-Schalter (*Bild 1*) auf 26 V erhöht und an das REPROM weitergegeben.

Nach Ablauf dieses Vorganges und Ablauf einer Zeit-Schleife erfolgt der gleiche Vorgang mit einer um 1 erhöhten Adresse von neuem, bis die programmierte Endadresse erreicht ist. In einer weiteren Schleife wird nun wiederum dieser Vorgang 144mal wiederholt, so daß die für den 2708 erforderliche Gesamtprogrammierimpulsdauer von 100 ms für das REPROM erreicht wird.

Die im *Bild 2* eingezeichneten Schalter S3 und S4 sind Adressierungsschalter; da ja immer nur eine *8-bit-Adresse* ausgegeben wird, muß die Umschaltung der Adressen 256-Byteweise von Hand erfolgen:

Adresse	S3	S4
0...255	offen	offen
256...511	offen	geschlossen
512...767	geschlossen	offen
768...1023	geschlossen	geschlossen

Der Schalter S2 legt +12 V an den CE/W-Eingang des REPROMs, die dort zur Programmierung notwendig sind. Ist er offen, so ist das REPROM zur Datenausgabe bereit, d. h. es werden Daten ausgegeben. Schalter S 1 legt den Programmierimpuls an den Eingang des EPROMs, er sollte genauso wie S 2 nur zum Programmieren geschlossen werden.

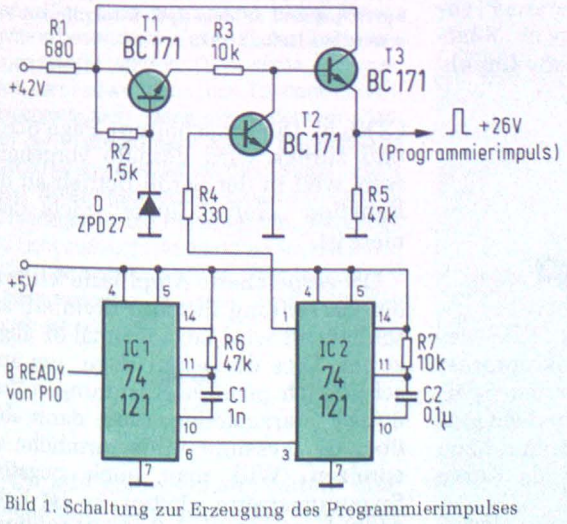


Bild 1. Schaltung zur Erzeugung des Programmierimpulses

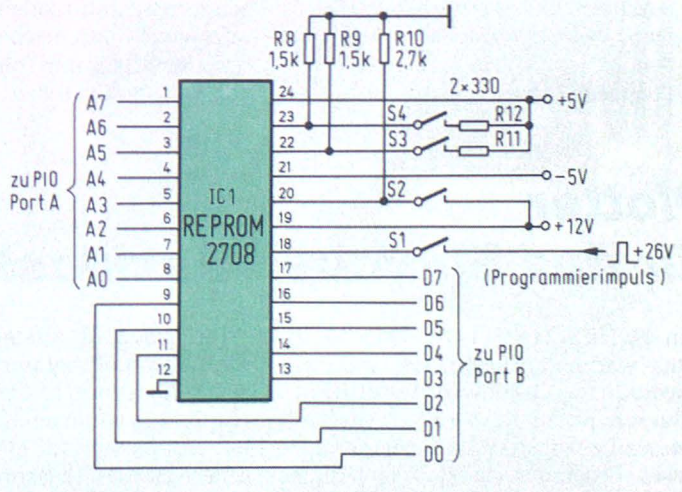


Bild 2. Anschluß eines REPROM-IC an ein Z-80-Kit

Auflistung des Maschinencodes für das Programmierprogramm:

Stack-Pointer 3CFFH
Startadresse 3C50H

3C50	0E90	3C65	1C
3C52	E5	3C66	15
3C53	D5	3C67	C25A3C
3C54	3E0F	3C6A	0D
3C56	D30A	3C6B	CC0D01
3C58	D30B	3C6E	D1
3C5A	7B	3C6F	E1
3C5B	D308	3C70	C3523C
3C5D	7E		
3C5E	D309		
3C60	06B4		
3C62	10FE		
3C64	23		

Beispiel einer Programmierung
(Z-80 KIT)

1. Programm eingeben
2. Daten eingeben (z. B. 1 KByte)
3. Adressenzähler auf Startadresse stellen (3C50H)
4. Anfangsadresse des abzuspeichernden Datenblocks in Reg HL eingeben (H = high Byte, l = low Byte, z. B. 4000H)
5. Teillänge +1 (≦ 256 Byte) in Reg. D eingeben.
6. Gewünschte Anfangsadresse des REPROM in Reg. E eingeben (z. B. 00)
7. Schalter S2 und S1 schließen und Schalter S3 und S4 für Adresse 0...0 FFH öffnen.
8. Programm starten.
9. Nach ca. 35 s erscheint die Adresse +1 des zuletzt ausgegebenen Daten-Bytes im Display.
10. Schalter S4 schließen (für die nächsten 256 Byte), d. h. REPROM-Adresse steht auf 0100H.
11. Reg. HL für neuen Teildatenblock laden (z. B. 4100H).
12. Teillänge +1 in Reg. D eingeben (z. B. 00 für 256 Byte).
13. Programm starten.
14. wie 9.
15. Schalter S4 öffnen, Schalter S3 schließen (für die nächsten 256 Byte), d. h. REPROM-Adresse auf 0200H.
16. wie 11. (z. B. 4200H,
17. wie 12.
18. wie 13.
19. wie 9.
20. Schalter S4 schließen (für die nächsten 256 Byte), d. h. REPROM-Adresse auf 0300H.
21. wie 11. (z. B. 4300H).
22. wie 12.
23. wie 13.
24. wie 9.
25. S1 und S2 öffnen, Programm-Ende.

Zum Schluß sei noch gesagt, daß es dem Benutzer überlassen bleibt, ein Leseprogramm für die Überprüfung des programmierten REPROM zu schreiben. Im übrigen kann das Programmiergerät selbstverständlich nicht nur für den Z-80-Kit verwendet werden, sondern auch für jeden anderen µP, sofern, wie beschrieben, zwei 8-bit-Schnittstellen und ein Ready-Impuls zur Verfügung stehen.

Startadresse des Programms ist 0120. Die Auslegung erfolgte für ein SWTPC-CT-64-Terminal, das 16 Zeilen mit je 64 Zeichen darstellen kann. Um vor dem Ausgeben der Kurve den Bildschirm zu löschen, enthält die Zeile 0119 einige ASCII-Zeichen, die folgender Terminal-Tastenbelegung entsprechen: Home Cursor = CTRL F; Erase Sreen = CTRL E; Page/Scroll Mode = CTRL N. Die Umschaltung Page/Scroll und zurück wird deshalb vorgenommen, weil in der Scroll-Betriebsart die Funktion „Home Cursor" nicht definiert ist.

```
0119 00 05 06 06 0E 0E 00
0120 A2 07     LDX #07     CURSOR
0122 BD 18 01  LDA 0118,X  HOME UND
0125 20 A0 1E  JSR 1EA0    BILDSCHIRM
0128 CA        DEX         LOESCHEN
0129 D0 F7     BNE 0122
012B A9 10     LDA #10     MAX.AMPL.
012D 85 F3     STA F3
012F C6 F3     DEC F3      AMPL.DEKR.
0131 10 03     BPL 0136
0133 4C 64 1C  JMP 1C64    ZUM KIM
0136 A9 00     LDA #00
0138 85 FA     STA FA      ZEIGER=
013A A9 02     LDA #02     0200
013C 85 FB     STA FB
013E A2 3F     LDX #3F     64 ZEICHEN
0140 A0 00     LDY #00     PRO ZEILE
0142 B1 FA     LDA (FA),Y
0144 4A        LSR A       AMPLITUDE
0145 4A        LSR A       DURCH
0146 4A        LSR A       8 DIVID.
0147 EA        NOP
0148 38        SEC         NULLPUNKT
0149 E9 0F     SBC #0F     VERSCHIEBEN
014B EA        NOP
014C C5 F3     CMP F3      VGL.ZEILE
014E F0 06     BEQ 0156    UND AMPL.
0150 20 9E 1E  JSR 1E9E    SPACE
0153 18        CLC
0154 90 05     BCC 015B
0156 A9 2E     LDA #2E     PUNKT
0158 20 A0 1E  JSR 1EA0    DRUCKEN
015B A0 08     LDY #08     ZEIGER
015D 20 63 1F  JSR 1F63    UM 8
0160 88        DEY         BYTE VOR-
0161 D0 FA     BNE 015D    RUECKEN
0163 CA        DEX
0164 D0 DA     BNE 0140
0166 20 2F 1E  JSR 1E2F    CRLF
0169 4C 2F 01  JMP 012F
016C 00        BRK         ENDE
```

6502-Programm zum Plotten einer gespeicherten Kurve, passend zu dem Speicheroszilloskop-Programm aus Heft 22/1978

Die gespeicherte Amplitude wird vor der Darstellung durch 8 dividiert; anschließend wird hexadezimal 0F abgezogen. Dies dient hier dazu, um ausschließlich positive Spannungen (hex 80...FF) darzustellen, diese dann aber über die gesamte Bildschirmhöhe zu spreizen. Will man auch negative Spannungswerte plotten, so ist Zeile 0147 durch 4A und Zeile 0149 durch E900 zu ersetzen. Fe.

Plotter für das Speicher-Oszilloskop

In FUNKSCHAU 1978, Heft 22, S. 1105, war beschrieben, wie man ein gewöhnliches Oszilloskop mit Hilfe des Mikrocomputers KIM-1 zu einem Speicheroszilloskop umfunktionieren kann. Dieses Programm bedient sich eines Analog-Digital-Wandlers, der die momentanen Amplitudenwerte in die Speicherzellen 0200...03FF ablegt.

Das im *Bild* aufgelistete KIM-Maschinencode-Programm (Mikroprozessor 6502) ist nun in der Lage, den Speicherinhalt, wenn auch nur in recht grober Auflösung, auf ein Terminal bzw. einen ASCII-Fernschreiber als Kurve auszugeben.

Diesen Vorgang nennt man „Plotten".

Hans Werner

Mikrocomputer

Eine kleine Einführung

10. Teil

Was bedeutet der Begriff „Assembler"?

- ☐ a) eine an den Maschinencode angelehnte Programmsprache. Die binäre Darstellung wird durch leicht merkbare Abkürzungen ersetzt.
- ☐ b) ein Übersetzungsprogramm, das maschinenorientierte Programmsprachen in den Maschinencode übersetzt.
- ☐ c) Maschinensprache eines Computers
- ☐ d) Objektprogramm eines Computers

bei Lösung
a) nach J
b) nach I
c) nach E
d) nach P

A

```
4 LA 31
5 UN 20
6 SP 31
7 LA 21
8 JP 3
```

Damit der Inhalt des Akkus auf jeden Fall ungleich 0 ist, wird zuerst die 1 (Zelle 21) in den Akku geladen.

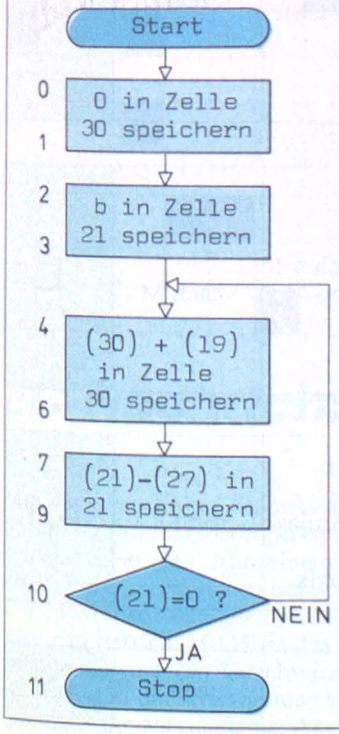
Richtige Mikrocomputer haben verschiedene Sprungbefehle. Bei ihnen ist dieser Kunstgriff überflüssig.

nach C

B

Das Flußdiagramm beschreibt ein einfaches Multiplikationsprogramm: $S = a \cdot b$
a und b sind ganze Zahlen, ungleich 0.
Die Multiplikation wird durch fortlaufende Addition gelöst, z. B.: $S = 2 \cdot 3 = 2 + 2 + 2 = 6$

Versuchen Sie die Lücken im Programm zu schließen. Eine kleine Hilfe: Zelle 21 wird als Zähler benützt, d. h. mit jedem Programmdurchlauf wird der Inhalt um 1 verringert, bis (b)mal addiert wurde.



Anfangsbedingung: S = 0

Anfangsbedingung: n = b

S = S+a (a+a+a..)

n = n-1 (b-1-1..)

Ist n = 0 ?

Speicherplatzbelegung:

Zelle 19: Faktor a
20: Faktor b
26: Zahl 0
27: Zahl 1
30: Ergebnis S

```
 0  LA 26   0 in den Akku laden
 1  SP 30   0 in Zelle 30 speichern
 2  .. ..
 3  .. ..
 4  LA 30   S in den Akku laden
 5  .. ..
 6  .. ..
 7  LA 21
 8  .. ..
 9  .. ..
10  JP 3    wenn (AC) ≠ 0: Sprung
11  ST      Stop
```

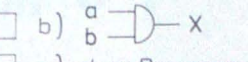
nach L

C

```
0  LA 20
1  UN 21
2  SP 22
3  LA 30
4  SU 22
5  SP 22
6  ST
```

Mit dem Programm werden zuerst eine logische Verknüpfung und dann eine Subtraktion ausgeführt. Welche Antwort beschreibt das Programm?
Zelle 20: Variable a, die nur 0 oder 1 sein kann
21: Variable b, die nur 0 oder 1 sein kann
22: Ergebnis X
30: Zahl 1

☐ a)

☐ b) 

☐ c) das Programm hat keinen Sinn.

bei Lösung
a) nach Q
b) nach H
c) nach N

D

Das ist leider nicht richtig. Versuchen Sie es nochmals.

nach A **E**

INTERFACE-Logik

nach G **F**

Schreiben Sie bitte ein kurzes Programm, durch das der Inhalt von Zelle 20 in den Akku gelangt.
Allerdings darf der Befehl LADEN (LA) nicht benützt werden.

nach O **G**

Die Aufgabe ist nicht einfach. Vielleicht haben Sie beim zweiten Versuch mehr Glück.

nach D **H**

Die Antwort ist richtig. Selbstverständlich trifft auch Lösung a zu. Maschinenorientierte Programmsprachen werden *auch* Assembler genannt.
Das Wort hat eine doppelte Bedeutung.

nach K **I**

Sehr gut. Natürlich ist auch Antwort b richtig, denn der Begriff Assembler wird für das eigentliche Programm und für das Übersetzungsprogramm benutzt.

nach K **J**

```
4  LA 31
5  UN 20
6  SP 31
7
8
```

Die Befehle 4...8 sollen ständig wiederholt werden.

Wie ist das zu machen, wenn in Zelle 28 eine 0 und in Zelle 21 eine 1 steht? Bitte die Befehle 7 und 8 eintragen.

nach B **K**

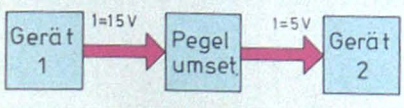
```
0  LA 26
1  SP 30
2  LA 20
3  SP 21
4  LA 30
5  AD 19
6  SP 30
7  LA 21
8  SU 27
9  SP 21
10 JP 3
11 ST
```

Wenn b = 0 ist, funktioniert das Programm nicht

Das kann durch eine Abfrage vor der Programmschleife behoben werden.

```
3  SP 21
   JP 3      b ≠ 0: nach 4
   ST        b = 0: Stop
4  LA 30
   usw.
```

nach M **L**



Zwei Geräte werden zusammengeschaltet. Gerät 1 hat die binären Ausgangspegel 15 V und 0 V. Für Gerät 2 sind aber 5 V und 0 V erforderlich. Die Anpassung übernimmt ein Pegelumsetzer.
Die allgemeine Bezeichnung dafür ist .....-Logik.

nach F **M**

Doch, das Programm hat einen Sinn. Versuchen Sie die Lösung zu finden.

nach D **N**

Lösung: SP 30 Speichern des Akkuinhalts in eine freie Zelle SU 30 Inhalt des Akkus 0 setzen: (30)–(30) = 0 AD 20 Inhalt der Zelle 20 in den Akku laden: 0 + (20) = (20)	nach D	O
Sie haben unrecht. Das Objektprogramm ist identisch mit der Maschinensprache und wird vom Computer (Objekt) direkt verstanden. Zurück zur Aufgabe.	nach A	P
Sehr gut. Die Subtraktion 1–N ergibt immer den negierten Wert $\overline{N}$, sofern N nur die Werte 1 oder 0 annehmen kann. N = 0: 1–0 = 1 N = 1: 1–1 = 0 Sollen alle 8 bit negiert werden, muß die 1 durch 255 ersetzt werden. Beispiel: 255 11111111 – N 01100101 $\overline{N}$ 10011010 Zum Glück ist der Test zu Ende.	PAUSE	Q

(Fortsetzung folgt)

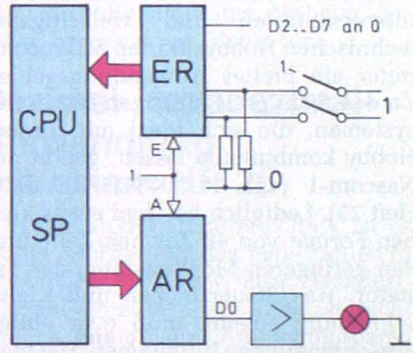
Tante Olgas Rätselecke

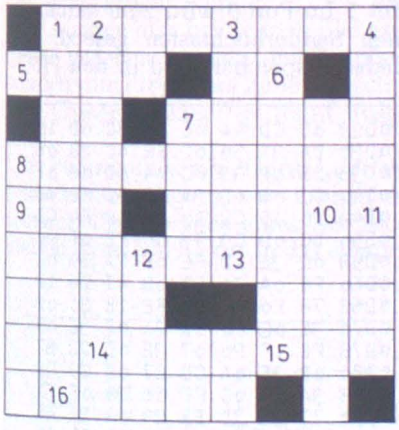
```
        AB   10
ANF     LA   31
        SP   25
        LA   DREI
        SU   25
        UN   25
        SP   31
        JP   ANF
DREI    NU   3
```

Mit welchem Schalter läßt sich die Lampe einschalten?

D0 = 1: Lampe brennt

Schalter offen: 0





Senkrecht:
1 Rechenart (Befehlsform)
2 auf ein P-Substrat aufgebaute MOS-Transistoren
3 $\overline{a} + \overline{b}$
4 an Tri-State-Ausgänge angeschlossene Datenleitungen
6 1024 bit
7 schnell arbeitende Logikfamilie
9 Steuerelektrode bei FETs
10 Zeit
11 Logikpegel nach DIN 41785
12 wichtige Einheit eines Mikrocomputers

Waagerecht:
2 Kurzzeichen für Zusammenzählen (Modellcomputer)
3 1 + 0 = 1
5 negiertes NAND
7 Hauptregister eines Computers
8 Sammelschiene für Daten
9 Abkürzung für Eingang
13 mittelschnelle und preiswerte Logikfamilie
14 Anschlußstift
15 große Leistungseinheit
16 Sprung

µC-Nachrichten

Im Rangierbahnhof Mannheim werden in Zukunft Mikrocomputer die Güterwagen über den Ablaufberg rollen lassen: Vor kurzem nahm man dort die erste mikroprozessorgesteuerte Gleisbremse in Betrieb. Mit Hilfe des Mikrocomputers wird die Geschwindigkeit der Wagen in der Gleisbremse so geregelt, daß die vorgegebene Auslaufgeschwindigkeit mit außerordentlich hoher Genauigkeit eingehalten werden kann. Dies nutzt man für die sogenannte Laufzielbremsung aus, die in der Mannheimer Anlage mit guten Ergebnissen erprobt worden ist. Die Wagen werden hier ohne Mitwirken von Hemmschuhlegern, die bei ihrer Arbeit sehr gefährdet sind, aber auch ohne die Investition teurer Beidrückeinrichtungen in den Richtungsgleisen „kuppelreif" gesammelt. Vor allem Rangierbahnhöfe mittlerer Größe lassen sich mit solchen mikrocomputergesteuerten Gleisbremsen und der Laufzielbremsung sehr wirtschaftlich automatisieren. Die Bremsensteuerung ist ein Baustein des von Siemens entwickelten Mikro-Computergesteuerten Rangiersystems **MICOR** für die Zugbildung und Zugverlegung in Rangierbahnhöfen.

Karl Schedler

RTTY-Betriebsprogramm für den Nascom-1

Das vorliegende Programm ermöglicht einen komfortablen Fernschreibbetrieb für Funkamateure (RTTY) im 8-bit-ASCII-Code. Es ist speziell für den Mikrocomputer Nascom-1 ausgelegt, benutzt also dessen Monitorunterprogramme.

Gerade im Amateurfunk als eines der interessantesten und vielseitigsten technischen Hobbys finden Mikrocomputer ein breites Anwendungsgebiet. Zu den recht preisgünstigen Prozessorsystemen, die sich ideal mit diesem Hobby kombinieren lassen, gehört der Nascom-1 (vgl. FUNKSCHAU 1978, Heft 25). Lediglich bei dem etwas kleinen Format von 48 Zeichen/Zeile und den geringeren Möglichkeiten der Tastatur, was Steuerzeichen und Kleinschreibung angeht, muß man einige Einschränkungen hinnehmen. Hat man allerdings einen erweiterten Speicherbereich zur Verfügung, so sind fast keine Grenzen gesetzt, solche fehlenden Funktionen selbst zu programmieren, und der Zeichengenerator kann ohnehin selbstverständlich alle 128 ASCII-Zeichen darstellen. Für die Steuerzeichen (hex 00 bis 1F) hat er graphische Symbole parat, die in diesem Programm aber unterdrückt werden.

Betrieb des Programms

Die Startadresse des Programms ist 0C60. Es besitzt drei mögliche Betriebsarten: Texteingabe (wird gleich nach dem Starten erreicht). Es erfolgt der Ausdruck „Text?". Jetzt kann ein max. 200 Zeichen umfassender Text eingetragen werden, der beim Senden jedesmal am Anfang automatisch ausgegeben wird. Die Tastenfunktionen in diesem Modus sind:

@ Ende des Textes, Sprung in den Empfangsmodus.

$ Ausgabe des im Wiederholspeicher stehenden Textes und dann Sprung in den Empfangsmodus.

Die „Überschrift"

Wenn der Cursor am rechten unteren Bildschirmrand angekommen ist und eine neue Zeile begonnen wird, rücken alle Zeilen um eine nach oben (scroll) – außer der obersten, sie dient nämlich als „Überschrift" zur Anzeige der Betriebsart (E = Empfang, S = Senden) und des Zustandes des FIFO-Pufferspeichers (L = Leer, F = Füllen, V = Voll). Außerdem kann das eigene Rufzeichen eingeblendet werden (*Bild 2*).

Empfangsbetrieb

Alle vom Modem kommenden ASCII-Zeichen zwischen hex 20 und hex 7F werden auf den Bildschirm geschrieben. Da der Nascom-1 nicht mit genormten Steuerzeichen (hex 00 bis hex 1F) arbeitet, müssen diese erst umcodiert werden, um die richtigen Funktionen auszulösen. Dies sind im einzelnen die Zeichen „Carriage Return", „Line Feed" und „Back Space". Als Funktionstasten im Empfangsmodus sind wirksam:
Shift 3 (£): Sprung in den Text-Eintragungsmodus.
@: Sender einschalten, Sendebetrieb.
Die anderen Tasten werden nicht abgefragt.

Sendebetrieb

Bit 5 im Port 0 wird zum automatischen Senderhochtasten gesetzt. Der Wiederholspeicher wird in den FIFO-

```
0C50 C7 A6 EF EE 07 06 CF EE
0C58 24 D4 EF EF E6 E6 FF EF
0C60 EF 54 65 78 74 3F 1F 00
0C68 21 00 0F 11 E0 0F CD 4D
0C70 0C 30 FB FE 40 28 1A FE
0C78 24 CA E1 0D CD 4A 0C FE
0C80 1D 20 03 2B 18 E7 CD 6C
0C88 0D 00 3A 55 0C FE 56 20
0C90 DD 36 00 CD 40 02 2A 18
0C98 0C E5 21 D0 0B 22 18 0C
0CA0 EF 44 43 20 32 20 4D 50
0CA8 41 20 20 52 54 54 59 20
0CB0 20 00 21 E6 0B 22 18 0C
0CB8 EF 46 49 46 4F 3A 20 20
0CC0 00 3E 45 32 E0 0B E1 22
0CC8 18 0C CD 4D 0C FE 23 CA
0CD0 60 0C FE 40 CA E6 0C DB
0CD8 02 17 30 EE DB 01 CD 55
0CE0 0D CD 4A 0C 18 E4 CD DC
0CE8 0D CD 40 02 21 E0 0B 36
0CF0 53 21 00 0E 11 00 0E D9
0CF8 21 00 0F 7E 23 FE 00 28
0D00 07 D9 CD 87 0D D9 18 F3
0D08 D9 06 FF 3A 55 0C FE 56
0D10 28 1D CD 35 00 CD 35 00
0D18 CD 4D 0C 38 07 DB 02 17
0D20 30 0D DB 01 FE 40 CA 43
0D28 0D CD 4A 0C CD 6C 0D 10
0D30 DA 3A 55 0C FE 4C 28 06
0D38 DB 02 87 FC A4 0D 06 01
0D40 C3 0B 0D 3A 55 0C FE 46
0D48 20 05 CD A0 0D 18 F4 CD
0D50 DC 0D C3 93 0C FE 08 20
0D58 02 3E 1D FE 0D CA B9 0D
0D60 FE 0A CA C1 0D 47 D6 1D
0D68 78 F0 AF C9 FE 1E 20 02
0D70 3E 0C FE 1D 20 02 3E 08
0D78 FE 1F 20 07 3E 0D CD 87
0D80 0D 3E 0A CD 87 0D C9 F5
0D88 3A 55 0C FE 56 28 0F F1
0D90 77 2C 7D BB 28 04 3E 46
0D98 18 02 3E 56 18 14 F1 C9
0DA0 DB 02 87 F0 1A D3 01 1C
0DA8 7B BD 20 04 3E 4C 18 02
0DB0 3E 46 32 ED 0B 32 55 0C
0DB8 C9 CD D0 0D 21 8A 0B 18
0DC0 0B CD D0 0D E5 CD F5 0D
0DC8 CD D0 0D E1 22 18 0C C9
0DD0 2A 18 0C 7E FE 5F 20 02
0DD8 36 20 AF C9 3E 20 C3 53
0DE0 00 21 00 0F 7E 2C FE 00
0DE8 CA 93 0C E5 CD 55 0D E1
0DF0 CD 4A 0C 18 EF F5 C5 D5
0DF8 E5 C3 95 01 00 00 00 00
```

Bild 1. Hexadezimale Auflistung des Programms. Von der FUNKSCHAU-Redaktion ist ein komplettes Assembler-Listing auf Anfrage erhältlich. Ab der Adresse 0CA1 sind 8 Bytes für das Rufzeichen einzuschreiben

Ausgabepuffer geladen und nicht mehr am Bildschirm angezeigt. Es kann in beliebiger Geschwindigkeit in den FIFO geschrieben werden. Nach einer kleinen Verzögerung (ca. 2 s) wird der FIFO-Speicher über den UART-Baustein ausgegeben.

Sollte der FIFO-Speicher voll sein, wird kein Zeichen mehr angenommen; man muß dann etwas warten, bis wieder einige Zeichen ausgegeben sind. Der Sendebetrieb kann durch Drücken der Taste a beendet werden. Der FIFO-Speicher wird dann noch vollständig geleert, der Sender ausgeschaltet und in den Empfangsmodus zurückgesprungen.

So sieht die „Überschrift" bei RTTY-Betrieb aus

Bemerkungen zum Programm

Es gibt drei Zeichen, die bestimmte Funktionen des Video-Monitorprogramms auslösen und leider aus unerfindlichen Gründen nicht mit den entsprechenden Standard-ASCII-Zeichen übereinstimmen. Dies sind:

Hex 1D: Cursor um 1 Stelle zurück (Back Space; Norm: 08).

Hex 1E: Bildschirm löschen (Form Feed; Norm: ØC)

Hex 1F: „Carriage Return" und „Line Feed" (0D und 0A).

Diese Zeichen werden in die jeweiligen Standard-Zeichen umgewandelt. Bei „1F" müssen zwei Zeichen (CR + LF) ausgegeben werden; „CR" und „LF" können also bei Senden nicht getrennt gegeben werden. Bei Empfang werden vom Programm Zeilenvorschub und Cursor-Rücklauf getrennt ausgeführt.

Im Programm kommen drei Arten der Bildschirmeintragung vor: Einmal die sequentielle Übergabe eines ASCII-codierten Bytes (im Akku) an das entsprechende Monitorprogramm, das es an die jeweilige Cursor-Position in der unteren Zeile einträgt; zweitens wie vorher, nur daß die Cursor-Position, die am RAM hinterlegt ist, per Programm zuerst an eine beliebige Stelle verlegt wird; und drittens, über einen Ladebefehl direkt-adressiert ins Video-RAM zu laden.

Das Programm belegt Speicherbereich 0C60 bis 0E00 (Bild 1), die übrigen Bereiche sind in FIFO- und Wiederholtextspeicher aufgeteilt. Der FIFO-Speicher ist großzügig ausgelegt (0E00...0EFF). Das hat programmiertechnisch zur Folge, daß man lediglich den niederwertigen Teil des FIFO-Ein- und Ausgabe-Zeigers „rotieren" lassen kann. Der Ausgabepointer hinkt dann dem Eingabepointer hinterher; man muß nur verhindern, daß die beiden „übereinanderstolpern".

Man kann so zu Testzwecken mit sehr geringen Geschwindigkeiten arbeiten, ohne beim Eintasten behindert zu sein.

Hardware-Zusätze

Der Nascom-1 kann in seiner Grundausstattung ohne Änderungen mit diesem Programm als Fernschreiber verwendet werden. Es ist lediglich noch ein Modem nötig, das eine V-24- bzw. RS-232-Schnittstelle besitzt. Für UKW genügen in den meisten Fällen einfache Schaltungen, wie sie z. B. in FUNKSCHAU 1978, Heft 14 und Heft 15, in der Rubrik „Schaltungskniffe" zu finden sind. Hier muß man nur noch kleine Anpassungen vornehmen; *Bild 3* zeigt die nötigen Zusätze, die beim Verfasser bereits ihren Dienst tun.

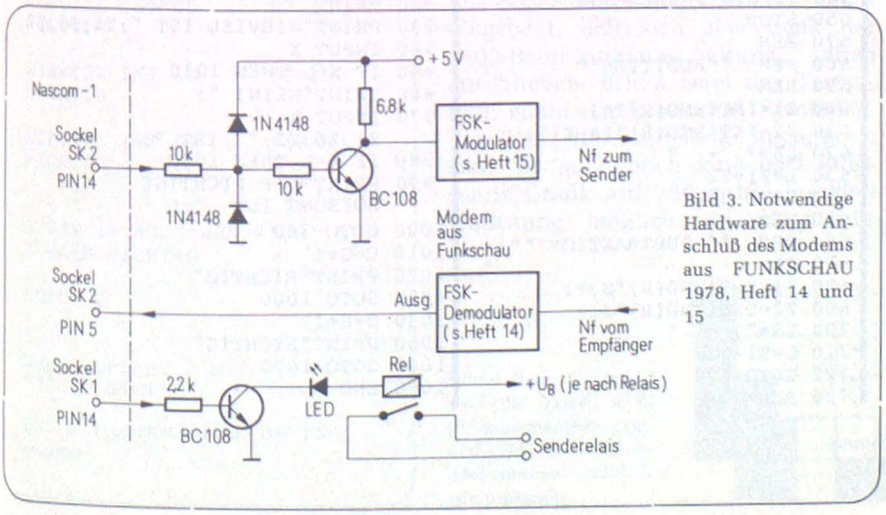


Bild 3. Notwendige Hardware zum Anschluß des Modems aus FUNKSCHAU 1978, Heft 14 und 15

Dezimal-Hexadezimal-Umwandlung mit einem HP 25

In der Mikrocomputertechnik ist es häufig erforderlich, dezimale Zahlen in ein binäres Byte bzw. eine Hexadezimalzahl umzurechnen. Dies ist recht einfach mit einem programmierbaren Taschenrechner möglich, der in umgekehrter polnischer Notation (UPN) arbeitet, z. B. mit den Typen HP-25 oder HP-25 C.

Tabelle: Programm für den HP-25

01	x ↔ y	11	x ↔ y
02	CLX	12	1
03	x ↔ y	13	+
04	1	14	GTO 03
05	5	15	R
06	f x ≧ y	16	EEX
07	GTO 15	17	2
08	–	18	÷
09	1	19	+
10	–	20	GTO 00

Die *Tabelle* zeigt die Programmauflistung, bestehend aus genau 20 Schritten. Nach Eingabe einer Dezimalzahl (0...255) und Drücken der Taste R/S erscheint in der Anzeige die gesuchte zweistellige Hexadezimalzahl, wobei das höherwertige „Nibble" vor dem Dezimalpunkt und das niederwertige hinter ihm steht. Die Anzeige „10.02" bedeutet z. B. hexadezimal A2. Leider ist die direkte Anzeige der Buchstaben A...F nicht möglich, daher dieser kleine Trick. *Fe.*

Peter Loster

Ein Rechenübungsprogramm mit ganzen Zahlen

Anlaß zu diesem BASIC-Programm gab ein kleiner Junge, der zwar gerne „Computer spielt", aber nicht gerne rechnet. Seitdem haben schon viele Kinder begeistert damit elementare Mathematik geübt.

Das vorliegende Programm (*Bild 1*) erzeugt elf Aufgaben in bis zu vier Grundrechenarten mit ganzen Zahlen. Die Grenzen können bei jeder Rechenart getrennt vorgegeben werden: Bei der Addition die maximale Summe und bei der Subtraktion der größte Minuend. Bei Multiplikation und Division wird eine Aufgabe aus dem gewählten „Einmaleins" z. B. bis 10 gestellt. Eine richtige Antwort wird mit einem Punkt bewertet, die richtige Antwort auf eine Wiederholungsfrage zählt einen halben Punkt. Nach 11 Aufgaben wird eine Bilanz ausgegeben und eine Note zwischen eins und sechs errechnet.

Das Programm wurde für das Cromemco-18-K-BASIC geschrieben. Der Verfasser bemühte sich aber, nur Standard-BASIC-Befehle zu verwenden. Lediglich bei den Stringvariablen können Unterschiede auftreten. Ersetzt man die Antworten (J/N) durch (0/1), so kann auf indizierte Stringvariable verzichtet werden.

Ist der Befehl in Zeile 60 nicht vorgesehen, so sollte die Variable R durch

```
60 PRINT „GIB IRGEND EINE ZAHL EIN"
65 INPUT R
```

```
 10 REM **RECHENUEBUNGSPROGRAMM**
 20 REM **GANZE ZAHLEN**
 30 REM
 40 REM  **EINGABE**
 50 REM
 60 RANDOMIZE
 70 PRINT
 80 PRINT
 90 PRINT"WELCHE RECHENARTEN (J/N)"
100 PRINT"  +  ";
110 INPUT A$(1)
120 PRINT"  -  ";
130 INPUT A$(2)
140 PRINT"  .  ";
150 INPUT A$(3)
160 PRINT"  :  ";
170 INPUT A$(4);
180 PRINT
190 PRINT"EINGABE DER GRENZEN: "
200 IF A$(1,1)="N"THEN 230
210 PRINT"ADDITION BIS";
220 INPUT A
230 IF A$(2,2)="N"THEN 260
240 PRINT"SUBTRAKTION BIS";
250 INPUT S
260 IF A$(3,3)="N"THEN 290
270 PRINT"MULTIPLIKANDEN BIS";
280 INPUT M
290 IF A$(4,4)="N"THEN 320
300 PRINT"DIVISION BIS";
310 INPUT D
320 REM
330 PRINT
340 PRINT
350 N=0 : G=0 : H=0
360 REM ** WAHL DER RECHENART **
370 N=N+1
380 W=INT(RND(R)*3.99)+1
390 IF A$(W,W)="N"THEN 380
400 IF N=12 THEN 430
410 ON W GOTO 570,650,730,810
420 GOTO 380
430 PRINT
440 PRINT
450 PRINT G;" VON 11 AUFGABEN
RICHTIG ";
460 IF H=0 THEN 480
470 PRINT H;" HALB RICHTIG "
480 PRINT
490 PRINT"NOTE  ";
    INT((11-G-H/2)/2)+1
500 PRINT
510 PRINT
520 PRINT"NEUE AUFGABE ";
530 INPUT B$
540 IF B$="J"THEN 330
550 STOP
570 REM
580 REM  **ADDITION**
590 REM
600 Z1=INT(RND(R)*A)
610 Z2=INT(RND(R)*(A-Z1))
620 Z$="  +  "
630 L=Z1+Z2
640 GOTO 890
650 REM
660 REM  ** SUBTRAKTION **
670 REM
680 Z1=INT(RND(R)*S)+1
690 Z2=INT(RND(R)*Z1)
700 Z$="  -  "
710 L=Z1-Z2
720 GOTO 890
730 REM
740 REM  ** MULTIPLIKATION **
750 REM
760 Z1=INT(RND(R)*M+0.5)
770 Z2=INT(RND(R)*M+0.5)
780 Z$="  .  "
790 L=Z1*Z2
800 GOTO 890
810 REM
820 REM  ** DIVISION **
830 REM
840 L=INT(RND(R)*D)+1
850 Z2=INT(RND(R)*D)+1
860 Z1=L*Z2
870 Z$="  :  "
880 GOTO 890
890 REM
900 REM  ** AUSGABE **
910 REM
920 PRINT
930 PRINT"WIEVIEL IST ";Z1;Z$;Z2;
940 INPUT X
950 IF X=L THEN 1010
960 PRINT"NEIN! ";
970 INPUT X
    Z1;Z$;Z2;"  IST ";
980 IF X=L THEN 1040
990 PRINT"DIE RICHTIGE
    LOESUNG IST  ";L
1000 GOTO 360
1010 G=G+1
1020 PRINT"RICHTIG"
1030 GOTO 1000
1040 H=H+1
1050 PRINT"RICHTIG"
1060 GOTO 1000
1080 END
```

Bild 1. Auflistung des BASIC-Rechenübungsprogramms

gesetzt werden. Dadurch werden nach dem Einschalten jedesmal andere Aufgaben erzeugt. Zeile 410 kann durch vier Zeilen IF... THEN... ersetzt werden.

Die Grundmenge läßt sich leicht auf negative Zahlen und Dezimalbrüche erweitern. Die Anzahl der Kommastellen kann ebenfalls mit der RND-Funktion variiert werden. Wechselnde Antworten beleben den Dialog. Zum Beispiel könnte bei der Wiederholungsfrage präzisiert werden: „Beachte die Zehnerstelle" o. ä. Zufällig und/oder vom Punktestand abhängig eingestreute Bemerkungen gestalten die Abfrage kurzweiliger und erhöhen die Bereitschaft, mehr zu üben.

Die hier abgedruckte Version verzichtet aus Gründen der Übersichtlichkeit auf diese Möglichkeiten, erfreut sich aber dennoch großer Beliebtheit (*Bild 2*).

```
 RUN

WELCHE RECHENARTEN (J/N)
 +  ? J
 -  ? N
 .  ? J
 :  ? J
EINGABE DER GRENZEN:
ADDITION BIS? 100
MULTIPLIKANDEN BIS? 10
DIVISION BIS? 10


WIEVIEL IST 13  +  36? 49
RICHTIG

WIEVIEL IST 0  .  9? 9
NEIN! 0  .  9  IST ? 1
DIE RICHTIGE LOESUNG IST  0

WIEVIEL IST 54  :  9? 6
RICHTIG

WIEVIEL IST 9  .  5? 45
RICHTIG

WIEVIEL IST 1  .  3? 1
NEIN! 1  .  3  IST ? 3
RICHTIG

WIEVIEL IST 3  :  1? 2
NEIN! 3  :  1  IST ? 3
RICHTIG

WIEVIEL IST 8  .  10? 80
RICHTIG

WIEVIEL IST 4  .  3? 12
RICHTIG

WIEVIEL IST 50  :  5? 10
RICHTIG

WIEVIEL IST 10  :  1? 10
RICHTIG

WIEVIEL IST 67  +  14? 81
RICHTIG


8 VON 11 AUFGABEN RICHTIG
2 HALB RICHTIG

NOTE  2


NEUE AUFGABE ? N
***550 STOP***
```

Bild 2. Typischer Ablauf des Programms

Heimcomputer aus Japan

Ab dem Frühjahr 1979 wird voraussichtlich auch aus Japan, nämlich von der Firma *Toshiba*, ein Heim- und Hobbycomputer zu haben sein. Er wird etwa so viel kosten wie der PET 2001, verfügt aber nicht über einen eingebauten Monitor, sondern erlaubt den Anschluß eines Farbfernsehgerätes. Damit entspricht er mehr dem schon bekannten Apple-II (vgl. FUNKSCHAU 1978, Heft 5, S. 174). Die Abmessungen sind dadurch natürlich wesentlich kompakter als die des PET (*Bild 1*).

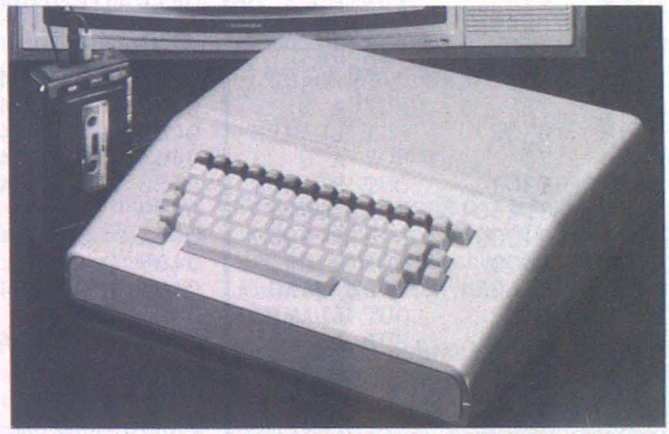


Bild 1. Das Äußere des Toshiba-Heimcomputers T-400 erinnert ein wenig an den Apple-II

Der Toshiba-Computer besitzt 8 K ROM mit dem BASIC-Interpreter und einem einfachen Monitorprogramm und in der Grundausstattung 4 K RAM. Ein Kassettenrecorder ist zwar nicht eingebaut, läßt sich aber dank der 1200-Baud-Interfaceschaltung außen anschließen. *Bild 2* zeigt das Tastenfeld; außer den grafischen Symbolen, die über die Shift-Taste zugänglich sind, ist auch noch eine hochauflösende Grafik mit 256 x 160 Punkten Auflösung möglich, allerdings nur schwarzweiß. In der Normalbetriebsart sind dagegen bis zu 16 Farben möglich.

Außer für den erwähnten Kassettenrecorder sind auch noch Anschlüsse für einen Drucker, für ein ASCII-Fernschreiber-Interface bzw. ein externes Terminal und „Joy Sticks" für Spiele vorhanden.

An später erhältlichen peripheren Geräten sind ein S-100-Bus-Adapter, ein Drucker, Joy Sticks, Floppy-Disk-Laufwerke und die für den Anschluß eines seriellen Fernschreibers erforderliche Interfaceschaltung vorgesehen.

Auch an die Belieferung der Kunden mit Software ist gedacht; Toshiba plant, Kassetten mit Assembler- und Disassembler-Programmen und Routinen für die Darstellung hochauflösender Grafiken sowie anderes Nützliches herauszubringen.

Welcher Mikroprozessor in dem T-400 von Toshiba steckt, war noch nicht zu erfahren; offenbar handelt es sich um ein japanisches Produkt mit 8 bit Wortlänge und 3 MHz Taktfrequenz.

Fe.

Bild 2. Beim Tastenfeld sind die Tasten RUN (Programmausführung) und LOAD (Kassette laden) ebenso unüblich wie praktisch

„Zahlen raten" — ein Spielprogramm

Das Programm wurde speziell für den ITT-Mikroprozessor-Experimenter geschrieben. Es kann jedoch auch auf jedem anderen 8080-System laufen, wenn die I/O-Ports entsprechend adressiert werden.

Bei diesem Spiel wird der Mikrocomputer als Spielpartner eingesetzt, d. h. man spielt gegen ihn. Die Regel dazu ist leicht erklärt: Zunächst „denkt" sich der Computer eine Zahl zwischen 0 und 99 (dez.). Der Spieler muß diese mit möglichst wenig Versuchen erraten. Dann wird der Spieß umgedreht: Der Spieler denkt sich eine Zahl, und der Computer darf sich den CPU-Kopf zerbrechen. Sieger ist, wer die wenigsten Versuche hat. Dieses Spiel läuft im Dezimalsystem, d. h. die Zahl 14 z. B. wird als „14" = 00010100 eingegeben bzw. angezeigt und nicht als „0E"!

```
0400  CDA004  CALL 04A0
0403  97      SUB A
0404  67      MOV H A
0405  D302    LLAMPE
0407  D301    RLAMPE
0409  CDAE04  CALL 04AE
040C  DB02    IN ASHALT
040E  93      SUB E
040F  DA5204  JC 0452
0412  C24D04  JNZ 044D
0415  CDC104  CALL 04C1
0418  7B      MOV A E
0419  D302    OUT LLAMPE
041B  7C      MOV A H
041C  D301    OUT RLAMPE
041E  CDB604  CALL 04B6
0421  010099  LXI B 9900
0424  2600    MVI H 00
0426  CDA004  CALL 04A0
0429  D302    OUT LLAMPE
042B  97      SUB A
042C  57      MOV D A
042D  D301    OUT RLAMPE
042F  CDC104  CALL 04C1
0432  CDAE04  CALL 04AE
0435  DB02    IN ASHALT
0437  FE02    CPI 02
0439  CA5F04  JZ 045F
043C  FE01    CPI 01
043E  CA8D04  JZ 048D
0441  97      SUB A
0442  D302    OUT LLAMPE
0444  7C      MOV A H
0445  D301    OUT RLAMPE
0447  CDB604  CALL 04B6
044A  C30004  JMP 0400
044D  3E02    MVI A 02
044F  C35404  JMP 0454
0452  3E01    MVI A 01
0454  D302    OUT LLAMPE
0456  CDC104  CALL 04C1
0459  CDB604  CALL 04B6
045C  C30C04  JMP 040C
045F  43      MOV B E
0460  78      MOV A B
0461  81      ADD C
0462  27      DAA
0463  6F      MOV L A
0464  D26804  JNC 0468
0467  14      INR D
0468  E5      PUSH H
0469  2600    MVI H 00
046B  7D      MOV A L
046C  C698    ADI 98
046E  27      DAA
046F  6F      MOV L A
0470  DA8304  JC 0483
0473  15      DCR D
0474  F28304  JP 0483
0477  C601    ADI 01
0479  27      DAA
047A  C28904  JNZ 0489
047D  CDC104  CALL 04C1
0480  C38904  JMP 0489
0483  CDC104  CALL 04C1
0486  C36B04  JMP 046B
0489  5C      MOV E H
048A  C39104  JMP 0491
048D  4B      MOV C E
048E  C36004  JMP 0460
0491  E1      POP H
0492  CDC104  CALL 04C1
0495  7B      MOV A E
0496  D302    OUT LLAMPE
0498  97      SUB A
0499  57      MOV D A
049A  CDB604  CALL 04B6
049D  C33504  JMP 0435
04A0  DB04    IN CSHALT
04A2  1C      INR E
04A3  E608    ANI 08
04A5  C2A004  JNZ 04A0
04A8  7B      MOV A E
04A9  C601    ADI 01
04AB  27      DAA
04AC  5F      MOV E A
04AD  C9      RET
04AE  DB04    IN CSHALT
04B0  E608    ANI 08
04B2  CAAE04  JZ 04AE
04B5  C9      RET
04B6  DB04    IN CSHALT
04B8  E608    ANI 08
04BA  C2B604  JNZ 04B6
04BD  CDAE04  CALL 04AE
04C0  C9      RET
04C1  7C      MOV A H
04C2  C601    ADI 01
04C4  27      DAA
04C5  67      MOV H A
04C6  C9      RET
```

Programmliste für das Spiel „Zahlen raten". Die I/O-Adressen beziehen sich auf den ITT-Mikroprozessor-Experimenter

Ablauf- und Bedienungsschema für den ITT-MP

1. Programm laden
2. Alle Schalter auf Null
3. Reset drücken
4. Mit „RUN" einmal takten. Der Mikrocomputer speichert eine Zufallszahl zwischen 0 und 99 (dez.). Auf der Anzeige erscheint „0000".
5. Der Spieler gibt nun eine beliebige Zahl mit dem A-Schalter ein.
6. „RUN" einmal takten. Auf den L-Lampen wird „02" angezeigt, wenn die eingegebene Zahl zu groß war (dann kleinere Zahl eingeben). Erscheint links eine „01", war die eingetastete Zahl zu klein (größere Zahl wählen).
7. 5. und 6. so lange wiederholen, bis auf dem linken Display die gespeicherte Zahl angezeigt wird. Auf den R-Lampen erscheint die Anzahl der Versuche.
8. Der Spieler denkt sich jetzt eine Zahl zwischen 0 und 99 (dez.).
9. „RUN" einmal takten. Auf den L-Lampen erscheint die geratene Zahl des Mikrocomputers.
10. War sein Wert zu groß, 02 mit den A-Schaltern eingeben, war er zu klein, dann „01" eintasten.
11. 9. und 10. so lange wiederholen, bis links die richtige Zahl erscheint. Dann „00" eingeben. Auf der R-Seite wird dann ebenfalls die Anzahl der Versuche, die der Mikrocomputer benötigte, angezeigt.
12. Ab 4. beliebig oft wiederholen.

Arno Ruff

Ein Plotter für den PET

Für den bekannten Mikrocomputer PET 2001 ist nun auch ein Plotter erhältlich. Die Krefelder Firma SVI entwickelte ein Interface, um den Hewlett-Packard-Plotter HP 7225 A an den PET anschließen zu können. In einer Sonderversion ist das Interface auch mit einem Zeichengenerator zum Erstellen von Text und Sonderzeichen erhältlich und kann dann neben dem Zeichnen von Kurven und Grafiken auch als Drucker verwendet werden. Die wichtigsten Befehle zur Ansteuerung sind Absolute Plot, Relative Plot, Pen Control, Formateinstellung und Textdarstellung. Allerdings: Der Plotter ist etwa dreimal teurer als ein PET.

Hans Werner

Mikrocomputer

Eine kleine Einführung

11. Teil

Wenn Sie die letzten Hefte verfolgt haben, so wird Ihnen aufgefallen sein, daß das verwendete Mikrocomputer-Modell so sehr vereinfacht war, daß ab und zu deutliche Unterschiede zu handelsüblichen Mikrocomputern erwähnt werden mußten. Wir führen deshalb hier ein neues Modell ein, das nun realen Prozessoren entspricht.

Modell 2

Unser neues Modell ist wiederum nach pädagogischen Gesichtspunkten aufgebaut. An ihm sollen die Eigenschaften handelsüblicher Mikrocomputer erklärt werden. Es zeichnet sich durch sinnvoll vereinfachte Steuersignale aus, so daß alle Interfaceschaltungen mit geringem Aufwand zu entwickeln sind. Der Befehlsvorrat entspricht fast einem wirklichen Mikrocomputer und erlaubt umfangreiche Programme.

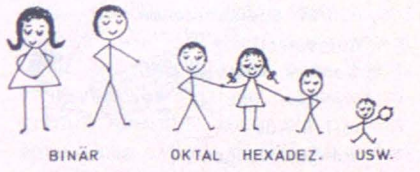
Mit Modell 2 werden Grundkenntnisse vermittelt, die den Übergang zur firmengebundenen Fachliteratur ermöglichen (z. B. 8008, 6800, F8).

Zusätzlich wird das Funktionsprinzip der wichtigsten Halbleiterspeicher besprochen.

Das Modell ist ein 8-bit-Computer. Für den Mikroprozessor ist 1 Byte nur ein Muster von 8 logischen Pegeln. Es ist die Aufgabe des Programmierers, den 8 Pegeln eine Bedeutung zuzuweisen.

1 Byte kann bedeuten:

1. Ein binäres Muster von 00000000...11111111.
2. Eine Binärzahl von 0...255.

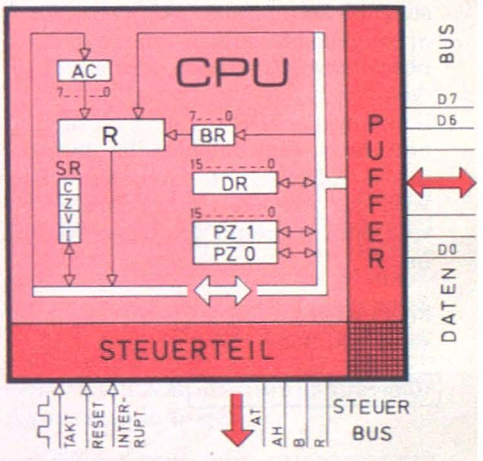


3. Eine binäre Zahl mit Vorzeichen von –128...+127. 1 Bit ist das Vorzeichen.
4. Eine Zahl von 0...99 im BCD-Code.
5. Eine Oktalzahl von 000...377.
6. Eine Hexadezimalzahl zwischen 00 und FF.
7. Ein Befehl (z. B. Addieren).
8. Die Hälfte einer Adresse. Die volle Adresse ist beim neuen Modell 16 bit lang.
9. Ein Zeichen im ASCII-Code.

Die CPU

Um das zu vermeiden, werden auch die schon bekannten Begriffe kurz wiederholt.

Die CPU ist die Zentraleinheit eines Mikrocomputers. In ihr werden die Daten verarbeitet. (CPU = Central Processing Unit)



Das Bild zeigt den internen Aufbau der CPU. Die wirkliche Ausführung kann von dieser Darstellung abweichen.

In den meisten Herstellerunterlagen ist der Inhalt der CPU als Programmiermodell wiedergegeben. Die tatsächliche interne Organisation ist viel zu unübersichtlich und für den Anwender vollkommen uninteressant.

Das ist auch ganz natürlich, denn für den Programmierer ist nur das Verhalten eines Gerätes wichtig. Die technische Realisierung ist für ihn ohne Bedeutung. Sogar der „Hardwaremann" interessiert sich lediglich für die Anschlußdaten und das Verhalten der CPU. Die CPU enthält zahlreiche Register, das Rechenwerk und den größten Teil der Steuerung.

Der Akkumulator AC
ist das Hauptregister der CPU. Bei allen

Operationen steht ein Operand im Akku.
Der Akkumulator wird oft als Brennpunkt der Datenbewegung bezeichnet.

Das Rechenwerk R
führt arithmetische, logische und Schiebebefehle aus. Nach der Operation wird das Ergebnis im Akku abgespeichert. Der Zwischenspeicher SP aus Modell 1 ist nicht eingezeichnet, weil er für den Programmierer ohne Interesse ist.

Der Programmzähler PZO
adressiert nach dem Einschalten der Stromversorgung den externen Speicher in der Reihenfolge 0, 1, 2 usw. In dieser Reihenfolge sind die Befehle im Speicher abgelegt. Sie werden nacheinander bearbeitet. Ausnahmen von dieser linearen Folge können durch Sprungbefehle erzielt werden. Dann übernimmt PZO den Inhalt des

Datenregisters DR
und setzt die Befehlsreihe mit der neuen Adresse fort. Im Datenregister wird der Adreßteil eines Befehls gespeichert, während der Operationsteil im

Befehlsregister BR
steht. Ein Befehl besteht aus 2 Teilen:

Operation	Adresse

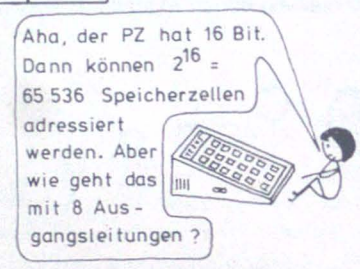
Bei bestimmten Befehlen wird statt der Adresse eine Konstante angegeben, z. B. die Zahl 5.

Das Statusregister
SR signalisiert den inneren Zustand der CPU (Übertragung, Vorzeichen, Ergebnis = 0, Interrupt).
Es besteht aus vier getrennten Flipflops und wird wie PZ1 später erläutert.

Die Speicherkapazität der Zähler und Register ist unterschiedlich groß.

AC	8 bit
BR	8 bit
DR	16 bit
PZ	16 bit

Bevor das erklärt wird, eine Frage. Wieviel kByte sind 2^{16} Bit?



Antwort: 65 536 : 1024 = 64 kByte oder $2^{16} : 2^{10} = 64$ kByte

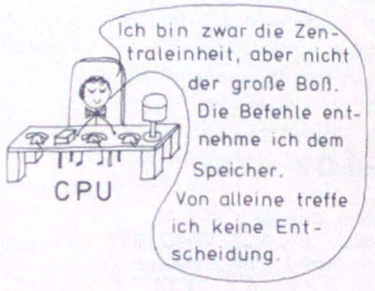
Ein kleines Programm soll in groben Zügen die Arbeit der CPU verdeutlichen. An die CPU ist ein Speicher angeschlossen, in dem das Programm in der richtigen Reihenfolge steht.

Programm
Zelle 0: Lade den Inhalt der Speicherzelle 100 in den Akku.

Zelle 1: Addiere den Inhalt des Akkus zum Inhalt der Speicherzelle 101.

Zelle 2: Speichere die Summe in Zelle 1000

Es soll also der Inhalt der Speicherzelle 100 zum Inhalt der Zelle 101 addiert und die Summe in Zelle 1000 gespeichert werden. Alle Datenbewegungen



erfolgen über den Akkumulator. Deshalb sind 3 Befehle notwendig. In der gleichen Reihenfolge sind die Befehle im Speicher abgelegt. Alle Befehle sind binär codiert und werden nacheinander von der CPU ausgeführt.

(Fortsetzung folgt)

Kassetten-Probleme beim KIM-1

Bei einer bestimmten Serie des Mikrocomputers KIM-1 haben die Kondensatoren C 9, C 10 und C 11 (Sollwert 6,8 nF) nur 680 pF. Erhebliche Schwierigkeiten beim Einlesen von Kassetten sind die Folge. Die fehlerhaften Kondensatoren tragen meist den Aufdruck „680/10". (Übrigens ist die Beschaltung des PLL-IC LM 565 im Schaltbild des KIM-1-Handbuches nicht vollständig eingetragen: Pin 1 liegt direkt und Pin 9 über 22 nF an Masse.)

Schwierigkeiten beim Einlesen von Kassetten, die nicht mit dem eigenen Recorder bespielt wurden, sind nahezu immer auf eine unterschiedliche Spaltjustage des Tonkopfes in den Recordern zurückzuführen. Empfehlenswert ist daher die Verwendung von Kassettengeräten, bei denen die „Taumelschraube" von außen z. B. durch ein kleines Loch zugänglich und während des Betriebes einstellbar ist. Dies geschieht ganz einfach auf beste Höhenwiedergabe.

Ab und zu treten auch Probleme auf, weil die verwendeten Kassetten nicht verwindungssteif sind oder andere mechanische Unstabilitäten aufweisen. Chromdioxid-Bänder bieten elektrisch keinerlei Vorteile gegenüber Eisenoxid-Bändern – zumindest bei der Datenaufzeichnung.

Noch ein Tip: Die Angabe im KIM-1-Handbuch, die obere Grenze des Applikation-RAM-Bereiches liege bei 17 EB, ist falsch. Vielmehr ist die letzte verwendbare Adresse 17 E6; die folgenden Zellen werden bereits vom Monitor-Programm verwendet. Fe.

Alles über Hobbycomputer...

...finden Sie in einem Sonderheft, das der Franzis-Verlag kürzlich unter dem Titel „Hobbycomputer" herausbrachte. Auf 138 Seiten im A-4-Format finden sich Beiträge über Programme in Maschinensprache und BASIC, Hinweise über Unterschiede verschiedener BASIC-Interpreter, Systembeschreibungen und allerlei nützliche Hardware-Tips. Besonders hervorgehoben seien eine vollständige hexadezimale Auflistung für ein 8080-BASIC, eine detaillierte Beschreibung der Monitor-Unterprogramme im ROM des KIM-1, eine Bauanleitung für ein Kassetten-Interface und eine Vielzahl von Anwendungsbeispielen für den PET 2001. Fachbeiträge über den S-100-Bus, über die Arbeitsweise von Floppy-Disk-Speichern und über Standardformate für die Datenaufzeichnung auf Lochstreifen und Magnetband runden den Inhalt ab. Das Sonderheft ist bei guten Buchhandlungen oder auch direkt beim Franzis-Verlag zu haben.

Dr. Hans-Jürgen Krupp

Ein einfacher 6502-Assembler

In Heft 21/1978 der FUNKSCHAU wurde ein einfacher 6502-Disassembler veröffentlicht. Wenn man die mit diesem Programm gegebenen Möglichkeiten wirklich ausschöpfen will, empfiehlt es sich, zumindest die im KIM-1 schon vorgesehene 4-K-Erweiterung vorzunehmen. Dann lohnt es sich aber auch, dieses Programm so zu erweitern, daß es wesentliche Assemblerfunktionen mit übernehmen kann.

Im folgenden wird ein Assembler/Disassembler vorgestellt, der insgesamt nur 1,25 K Speicherplatz benötigt. Außerdem werden in der „Zero Page" Arbeitsspeicher benötigt, und zwar der Bereich von 00B0 bis 00EE und zusätzlich so viele Speicherplätze, wie Sprungmarken definiert werden.

Die Funktionen des Assemblers

Selbstverständlich kann man von einem derart kleinen Assembler nicht allzu viel erwarten. Ein voll „ausgebauter" Assembler würde aber die erste 4-K-Erweiterung voll in Anspruch nehmen. Das hier vorgestellte Assemblerprogramm unterliegt dementsprechend einigen Einschränkungen. Die wichtigsten sind:

1. Es ist nicht möglich, beliebige Variablennamen zu benutzen. Als Variablennamen werden Speicheradressen benutzt. Die Zuordnung von Speicheradressen zu Variablennamen muß also durch den Programmierer erfolgen.
2. Auch als Sprungmarken sind nicht beliebige Namen benutzbar. Die Sprungmarken beginnen vielmehr mit dem Buchstaben „M" und einer sich anschließenden zweistelligen hexadezimalen Zahl.
3. Wie in vielen Assemblern wird auch hier ein festes Eingabeformat verwendet, d. h. es kommt auf die Stellung des einzelnen Zeichens für die korrekte Interpretation an.
4. Unzulässige Befehle und Adressierungsarten führen nicht unbedingt zu einer Fehlermeldung. Sie sind allerdings während der Eingabe erkennbar, da der erzeugte Code mit Hilfe des Disassemblers sofort umgesetzt und wieder ausgegeben wird. Insofern könnte man dieses Programm auch als „Dialogprogramm" ansehen, da es auf die Eingabe eines einzelnen Befehls sofort mit der Ausgabe des erzeugten hexadezimalen und wieder disassemblierten Codes reagiert.
5. In der beschriebenen Version erfolgt die Eingabe nur über eine ASCII-Tastatur (TTY/Terminal) und nicht von Band. Das Programm ist jedoch so konzipiert, daß es ohne Schwierigkeiten mit einem Editor verbunden werden könnte und die Eingabe auch von Band erfolgen kann. Damit wären insbesondere Programmverbesserungen erheblich leichter möglich.

Auch wenn diese Einschränkungen durchaus bedeutsam sind, erleichtert der hier vorgestellte Assembler die Programmierarbeit erheblich. Es ist insbesondere nicht nötig, jeweils den zu einem Assemblerbefehl gehörigen Maschinencode nachzuschlagen. Außerdem übernimmt das Programm alle Adressenrechnungen.

Die Eingabe der Assemblerbefehle

Generell gilt die Grundregel, daß (mit einer Ausnahme, die später noch behandelt wird) die Eingabe genauso erfolgen muß, wie die Ausgabe des Disassemblers gestaltet ist. In Zweifelsfällen kann man sich einfach damit helfen, daß man einen bestimmten Befehl in Maschinencode eingibt und durch den Disassembler ausgeben läßt. Diese Ausgabe kann man dann in Zukunft als Muster der Eingabe benutzen. In Bild 2 von Heft 21/1978 finden sich übrigens zahlreiche Beispiele für das Eingabeformat.

Die Abkürzungen für die einzelnen Instruktionen entsprechen der MOS-Technology-6502-Literatur. Sie belegen die ersten drei Zeichenfelder des Eingabeformats. Das vierte Feld ist in der Regel eine Leerspalte; lediglich für den Fall des indirekten Sprunges steht hier eine Klammer. Das fünfte und sechste Feld enthält gegebenenfalls die erste Hälfte der Adresse (MSB), bei Zero-Page-Adressierung ist es leer. Das siebente und achte Feld enthält den zweiten Teil (LSB) der Adresse. Bei beiden Formen der indirekten Adressierung, die sich ja jeweils auf die Zero-Page bezieht, werden diese Felder eingeklammert; anschließend folgen ein Komma und der jeweilige Index, also ein X oder ein Y. An dieser Stelle unterscheidet sich die Eingabe ausnahmsweise von der Ausgabe: Die indizierte indirekte Adressierung mit Hilfe des X-Index wird genauso eingegeben wie die indirekte indizierte Adressierung mit Hilfe des Y-Index.

Bei der einfachen indizierten Adressierung folgen der zweiten Hälfte der Adresse zunächst in Feld 9 ein Komma und in Feld 10 der Index (X oder Y).

Die unmittelbare Adressierung kann nur hexadezimal erfolgen. Sie wird eingeleitet durch die Zeichen # $ im fünften und sechsten Feld. Die Akkumulatoradressierung ist durch ein „A" im sechsten Feld gekennzeichnet.

Bei Sprunganweisungen steht die Sprungmarke, eingeleitet durch ein #, im Adressenfeld, d. h. in Spalte 6 ein M

und in den Spalten 7 und 8 eine zweistellige Hexazahl. Diese Sprungmarken werden übrigens in der ersten Assemblerphase falsch interpretiert. Hierdurch darf man sich nicht beirren lassen; im erzeugten Code wird zunächst gerade diese Hexazahl ausgewiesen. Sie wird erst in der zweiten Phase in eine relative Adresse umgerechnet.

Ein Abweichen von dieser Reihenfolge oder die Verschiebung eines Zeichens auch nur um eine Spalte führt in der Regel zu einer fehlerhaften Assemblierung.

Jede einzelne Instruktionszeile wird durch einen Punkt beendet. Dieser kann an beliebiger Stelle stehen, er löst die Erzeugung des Maschinencodes und die Disassemblierung aus.

Normalerweise beginnt eine Instruktion in der ersten Spalte. In einigen Sonderfällen beginnt die Instruktion erst in der 5. Spalte; in der ersten Spalte steht dann eines der Zeichen F, H, M oder Z, in den Spalten 2 und 3 Hexazeichen, in der Spalte 4 ein Leerzeichen.

Durch ein **H** kann ein beliebiges hexadezimal verschlüsseltes Wort eingegeben werden. Es wird in der ersten Assemblerphase nicht analysiert, so daß auf diese Art und Weise auch Tabellenfelder mit Hilfe des Assemblers eingegeben werden können.

Der Buchstabe **M** ist der Beginn einer Sprungmarke. Es folgt eine zweistellige Hexazahl. Diese bestimmt zugleich die Speicheradresse in der Zero Page. Es empfiehlt sich daher, eine möglichst niedrige Zahl zu verwenden. Eine höhere Zahl als AF ist auf jeden Fall unzulässig.

Durch den Buchstaben **Z** ist es möglich, eine beliebige Zeile anzuspringen[1]). Die dem Z folgenden zwei Zeichen geben die zweite Hälfte der Adresse an, in der die folgende Instruktion gespeichert werden soll. Ein Überspringen der Page ist allerdings nicht möglich. Der Befehl ist vor allem für Korrekturen gedacht, die übrigens nicht mit „Backspace" erfolgen können.

Der Buchstabe **F** bewirkt die Beendigung der ersten Phase der Assemblierung. In dieser Phase sind ja bei Sprungbefehlen nicht tatsächliche Adressen, sondern Sprungmarken angegeben. Diese müssen nun noch in relative Adressen umgerechnet werden. Da die hier in der ersten Phase erzeugten Sprungadressen noch unkorrekt sind, ist eine zweite und jetzt endgültige Ausgabe des Programms notwendig; sie enthält die korrekten Adressen.

```
0C00 A2   0C46 1E   0CA8 C4   0D38 E8
0C01 01   0C47 0E   0CA9 00   0D39 0C
0C02 86   0C4A 0F   0CB3 C8   0D44 0D
0C03 CE   0C4E 0C   0CB4 00   0D49 0D
0C04 20   0C53 0F   0CBC C2   0D73 0D
0C05 DD   0C64 0D   0CBD 00   0D75 C6
0C06 0C   0C66 C3   0CC1 C7   0D76 00
0C07 EA   0C67 00   0CC2 00   0D78 C5
0C08 EA   0C69 C4   0CCE 0E   0D79 00
0C09 EA   0C6A 00   0CD6 0D   0D7E 0D
0C0B 1A   0C71 0D   0CD9 0D   0D80 C3
0C0F AF   0C73 C5   0CDB D0   0D81 00
0C10 00   0C74 00   0CDC 0D   0D84 0C
0C18 0D   0C76 C6   0D02 0E   0D8D 0F
0C1A B0   0C77 00   0D04 BF   0D90 0F
0C1B 00   0C7E C3   0D05 00   0D97 1E
0C1D B1   0C7F 00   0D08 0F   0D98 0E
0C1E 00   0C8B C7   0D0A C0   0D9B 0F
0C23 0D   0C8C 00   0D0B 00   0D9D BF
0C25 B2   0C90 C8   0D0E 0F   0D9E 00
0C26 00   0C91 00   0D10 C1   0DA1 0F
0C28 B3   0C99 C7   0D11 00   0DA3 C0
0C29 00   0C9A 00   0D13 F0   0DA4 00
0C35 0D   0C9E C8   0D14 0D   0DA7 0F
0C38 0D   0C9F 00   0D18 B0   0DA9 C1
0C43 C9   0CA3 C9   0D19 00   0DAA 00
0C44 00   0CA4 00   0D1F 1A   0DAD 0D
```

Bild 1. Notwendige Änderungen des in den Bereich 0C00...0FFB verschobenen Disassembler-Programms aus Heft 21/1978

Die Wirkungsweise des Programms

Das Programm orientiert sich an der Struktur des zugrundeliegenden, schon erwähnten Disassemblers. In einem ersten Schritt wird dieser um die Ausgabe des Maschinencodes erweitert (0DF0 bis 0E1D). Im Assembler werden zunächst die Zeichen F, H, M oder Z ausgewertet (1016 bis 1059). Nach dem Einlesen des Eingabepuffers (105A bis 1069) wird zunächst festgestellt, ob es sich um einen Befehl mit einer einzigen Adressierungsart handelt (106A bis 1097). Andere Befehle werden bestimmt (1098 bis 10C2), die Adressierungsart wird festgestellt und der entsprechende Maschinencode erzeugt (0E2F bis 0EBA). Der gefundene Code und ggf. die dazugehörigen Adressen werden abgespeichert (10C3 bis 10FC). Anschließend wird der Disassembler aufgerufen, der in der ersten Phase nach jeder Zeile zum Assembler zurückspringt. In der durch den Buchstaben „F" ausgelösten zweiten Phase wird nur der Disassembler angesprochen; dabei werden jeweils die relativen Sprungadressen berechnet (0DD0 bis 0DE7).

Die Übernahme auf den KIM

Da das Gesamtprogramm 1,25 K Speicherplatz benötigt, empfiehlt es sich, das Programm im Erweiterungsbereich abzuspeichern. In unserem Beispiel ist hierfür der Speicherplatz von 0C00 bis 10FF vorgesehen. Das Programm kann wie folgt geladen werden: Ausgangspunkt ist der in Bild 3 von Heft 21/1978 abgedruckte Code. Er wird in die Zellen 0000 bis 03FB eingegeben. Durch ein Verschiebungsprogramm (z. B. MOVIT aus dem „First Book of KIM") oder durch Überspielen auf Band mit ID = FF wird dieser Code in den Bereich 0C00 bis 0FFB verschoben. Die in *Bild 1* angesprochenen Zellen müssen entsprechend den hier enthaltenen Angaben verbessert werden. Der in *Bild 2* enthaltene zusätzliche Code ist in die entsprechenden Speicherzellen einzugeben. Damit ist das Programm geladen.

Als nächstes sind die Start- und Endadressen des zu bearbeitenden Programms in derselben Art wie beim Bandbeschreiben in die Zellen 17F5 bis 17F8 einzugeben. Soll das Programm nur als Disassembler benutzt werden,

[1]) Normalerweise ist es nicht möglich, sofort nach dem Z-Befehl eine Programm-Marke (M) zu setzen. Ersetzt man E9 an der Adresse 1059 durch AD, so wird nach ZXX eine neue Zeile begonnen, und es ist wieder möglich, danach eine Marke zu setzen.

```
/0CDD/AD F5 17 85 /0CE1/FA AD F6 17 /0CE5/85 FB 60 A5 /0CE9/CE D0 03 4C
/0CED/07 10 4C 0A /0CF1/0C A5 FA 8D /0CF5/F7 17 A5 FB /0CF9/8D F8 17 4C
/0CFD/04 0C EA

/0DD0/A5 CE C9 02 /0DD4/D0 0F A0 01 /0DD8/B1 FA AA B5 /0DDC/00 18 E5 FA
/0DE0/38 E9 01 91 /0DE4/FA 4C 57 0D /0DE8/EA EA EA EA /0DEC/EA EA EA EA
/0DF0/20 2F 1E A0 /0DF4/00 84 EE A2 /0DF8/00 86 ED A6 /0DFC/F4 86 EC B1
/0E00/FA 20 3D 0D /0E04/A6 ED 9D B6 /0E08/00 98 9D B5 /0E0C/00 E8 E8 E8
/0E10/86 ED C6 EC /0E14/F0 07 A4 EE /0E18/C8 84 EE D0 /0E1C/E2 60 A2 01
/0E20/B6 F4 A2 06 /0E24/A9 46 9D C2 /0E28/00 CA D0 FA /0E2C/4C 12 0D A5
/0E30/CF 85 CD 29 /0E34/0F A6 D5 E0 /0E38/23 D0 08 C9 /0E3C/01 D0 4C A9
/0E40/0B D0 72 C9 /0E44/03 10 0B A5 /0E48/CF C9 20 F0 /0E4C/6D 18 69 04
/0E50/85 CF A6 D4 /0E54/E0 28 D0 04 /0E58/A9 6C D0 5C /0E5C/C9 0C F0 2B
/0E60/A5 D5 C9 20 /0E64/D0 31 A4 D6 /0E68/C0 20 D0 0D /0E6C/A5 CF A6 D9
/0E70/E0 2C D0 44 /0E74/18 69 10 D0 /0E78/3F C0 41 D0 /0E7C/04 A9 04 D0
/0E80/34 C0 28 D0 /0E84/38 A5 DB C9 /0E88/58 D0 04 A5 /0E8C/CD D0 29 C9
/0E90/59 D0 2A A9 /0E94/0C D0 1E A5 /0E98/DA C9 20 D0 /0E9C/04 A9 08 D0
/0EA0/14 C9 58 D0 /0EA4/04 A9 18 D0 /0EA8/0C C9 59 D0 /0EAC/10 A9 14 A6
/0EB0/CD E0 A2 F0 /0EB4/F0 18 65 CF /0EB8/85 CF 4C C3 /0EBC/10 A2 0B A9
/0EC0/46 20 A0 1E /0EC4/CA D0 F8 4C /0EC8/0A 0C EA EA /0ECC/EA EA EA EA
/0ED0/EA EA EA EA /0ED4/EA EA EA EA /0ED8/EA EA EA EA /0EDC/EA EA EA EA

/1000/20 DD 0C A2 /1004/00 86 CE A2 /1008/0E A9 20 95 /100C/D1 CA D0 F9
/1010/20 2F 1E 20 /1014/5A 1E C9 46 /1018/D0 0A A2 02 /101C/86 CE 4C F2
/1020/0C 4D 48 5A /1024/A2 03 DD 20 /1028/10 F0 05 CA /102C/D0 F8 F0 2A
/1030/EA EA EA 20 /1034/9D 1F E0 03 /1038/F0 1B E0 02 /103C/F0 0D AA A5
/1040/FA 95 00 20 /1044/5A 1E 20 5A /1048/1E D0 0F A0 /104C/00 91 FA 20
/1050/63 1F 18 90 /1054/B2 85 FA 18 /1058/90 E9 85 D1 /105C/A2 01 20 5A
/1060/1E C9 2E F0 /1064/05 95 D1 E8 /1068/D0 F4 86 D0 /106C/A2 00 A5 D1
/1070/DD 00 0F F0 /1074/07 E8 E0 20 /1078/D0 F4 F0 1C /107C/A5 D2 DD 20
/1080/0F F0 02 D0 /1084/F0 A5 D3 DD /1088/40 0F F0 02 /108C/D0 E7 BD E0
/1090/0E A0 00 91 /1094/FA 18 90 2B /1098/A2 00 A5 D1 /109C/DD 60 0F F0
/10A0/0B E8 E0 17 /10A4/D0 F4 4C BD /10A8/0E A5 D2 DD /10AC/78 0F F0 02
/10B0/D0 EF A5 D3 /10B4/DD 90 0F F0 /10B8/02 D0 E6 BD /10BC/C0 0F 85 CF
/10C0/4C 2F 0E A6 /10C4/FA 86 CA A6 /10C8/FB 86 CB A0 /10CC/00 A2 02 91
/10D0/FA B5 D5 C9 /10D4/20 F0 1D C9 /10D8/23 F0 19 20 /10DC/AC 1F B5 D6
/10E0/20 AC 1F A5 /10E4/F8 20 63 1F /10E8/A0 00 91 FA /10EC/E0 00 F0 04
/10F0/CA CA F0 DD /10F4/A5 CA 85 FA /10F8/A5 CB 85 FB /10FC/4C 0A 0C 00
```

Bild 2. Zusätzliche Programme des Assemblers. Die Assembler-Startadresse ist 1000, der Disassembler kann an der Adresse 0C00 gestartet werden

ist es an der Adresse 0C00 zu starten. Soll das Programm als Assembler benutzt werden, ist die Startadresse 1000. Jetzt können die gewünschten mnemonischen Befehle eingegeben werden.

Die Eingabe wird durch den Buchstaben F beendet. Hierdurch wird die zweite Phase des Assemblers aufgerufen und das fertig assemblierte Programm ausgedruckt. Außerdem wird die Endadresse in 17F7 und 17F8 an die Länge des Programms angepaßt.

Ein Beispiel

Bild 3 zeigt ein Beispiel. Hier wird der Maschinencode für ein einfaches Programm erzeugt, mit dessen Hilfe der Speicherinhalt eines bestimmten Bereichs ausgedruckt werden kann. Im Gegensatz zum KIM-Monitor werden hier die einzelnen Bytes durch ein Leerzeichen getrennt. Eingegeben werden jeweils die kurzen Zeilen, die langen Zeilen werden vom Assembler erzeugt.

Ein Beispiel für die Verwendung von Z zeigt die Zeile 030A. Hier vertippte sich der Programmierer; der erzeugte Code war daher falsch. Mit Hilfe der Eingabe von Z0A wird diese Zeile erneut eingegeben.

Nur der Vollständigkeit halber findet sich am Ende des Programms ein Beispiel für die Verwendung des Zeichens „H". Mit seiner Hilfe wird am Schluß des Programms das Hexazeichen „FF" eingegeben, das konsequenterweise bei der Assemblierung in der zweiten Phase zu einer Fehlermeldung führt. Die Verwendung der Sprungmarken wird aus dem Beispiel wohl ausreichend deutlich. Das erzeugte Programm könnte jetzt durch Anspringen der Adresse 0300 unmittelbar ausgegeben werden.

Auch wenn es sich hier nicht gerade um einen luxuriösen Assembler handelt, zeigt das vorliegende Beispiel deutlich, daß die Programmierarbeit mit Hilfe dieses einfachen Assemblers wesentlich erleichtert werden kann.

```
17F5 00
17F6 03
17F7 FF
17F8 03 1000
1000 20 G
JSR 0CDD.
0300 20 DD 0C   JSR 0CDD
M02 LDX #$10.
0303 A2 10      LDX #$10
STX   CB.
0305 86 CB      STX   CB
JSR 1E2F.
0307 20 2F 1E   JSR 1E2F
JSR 1E1F.
030A 20 1F 1E   JSR 1E1F
Z0A JSR 1E1E.
030A 20 1E 1E   JSR 1E1E
LDA #$2F.
030D A9 2F      LDA #$2F
JSR 1EA0.
030F 20 A0 1E   JSR 1EA0
M01 LDY #$00.
0312 A0 00      LDY #$00
LDA  (FA),Y.
0314 B1 FA      LDA  (FA),Y
JSR 1E3B.
0316 20 3B 1E   JSR 1E3B
JSR 1E9E.
0319 20 9E 1E   JSR 1E9E
LDA   FA.
031C A5 FA      LDA   FA
CMP 17F7.
031E CD F7 17   CMP 17F7
BNE #M00.
0321 D0 00      BNE 0323
LDA   FB.
0323 A5 FB      LDA   FB
CMP 17F8.
0325 CD F8 17   CMP 17F8
BNE #M00.
0328 D0 00      BNE 032A
JMP 1C4F.
032A 4C 4F 1C   JMP 1C4F
M00 JSR 1F63.
032D 20 63 1F   JSR 1F63
DEC   CB.
0330 C6 CB      DEC   CB
BNE #M01.
0332 D0 01      BNE 0335
BEQ #M02.
0334 F0 02      BEQ 0338
HFF
F
0300 20 DD 0C   JSR 0CDD
0303 A2 10      LDX #$10
0305 86 CB      STX   CB
0307 20 2F 1E   JSR 1E2F
030A 20 1E 1E   JSR 1E1E
030D A9 2F      LDA #$2F
030F 20 A0 1E   JSR 1EA0
0312 A0 00      LDY #$00
0314 B1 FA      LDA  (FA),Y
0316 20 3B 1E   JSR 1E3B
0319 20 9E 1E   JSR 1E9E
031C A5 FA      LDA   FA
031E CD F7 17   CMP 17F7
0321 D0 0A      BNE 032D
0323 A5 FB      LDA   FB
0325 CD F8 17   CMP 17F8
0328 D0 03      BNE 032D
032A 4C 4F 1C   JMP 1C4F
032D 20 63 1F   JSR 1F63
0330 C6 CB      DEC   CB
0332 D0 DE      BNE 0312
0334 F0 CD      BEQ 0303
0336 FF           FFFFFF
KIM
0337 C0
```

Bild 3. Die beiden Assembler-Phasen am Beispiel eines einfachen Dump-Programms, mit dem auch Bild 2 zustandekam

Druckerprogramm für den µP 6502

Für weniger als 400 DM bekommt man heute schon einfache Matrixdrukker, die mit aluminiumbeschichtetem Papier arbeiten; ein Beispiel ist z. B. der von Matsushita hergestellte KE-100. Er benötigt zwei Versorgungsspannungen (−24 V für die Druckköpfe und +5 V für das IC) und arbeitet im 6-bit-ASCII-Code; Kleinbuchstaben können daher nicht dargestellt werden, aber für die Anfertigung von Programmauflistungen oder ähnliche Aufgaben spielt das ja auch keine Rolle.

```
6502-DRUCKERPROGRAMM
0000 A9 FF    LDA #$FF
0002 8D 01 17 STA $1701
0005 A9 02    LDA #$02
0007 8D 03 17 STA $1703
000A A2 00    LDX #$00
000C 8E 02 17 STX $1702
000F 8D 02 17 STA $1702
0012 F0 0F    BEQ 0023
0014 E8       INX
0015 E0 10    CPX #$10
0017 D0 03    BNE 001C
0019 4C 4F 1C JMP $1C4F
001C AD 02 17 LDA $1702
001F 29 04    AND #$04
0021 F0 F9    BEQ 001C
0023 BD D0 01 LDA $01D0,X
0026 8D 00 17 STA $1700
0029 AD 02 17 LDA $1702
002C 29 04    AND #$04
002E D0 F9    BNE 0029
0030 F0 E2    BEQ 0014
```

Bild 1. Programm zum Ausdrucken einer aus 16 Zeichen bestehenden Zeile. Die Branch-Sprünge wurden hier in absolute Adressen umgerechnet

Der Anschluß an den Mikrocomputer KIM-1 oder ein anderes System auf 6502-Basis erfordert nur einen geringen Programmaufwand. Das in *Bild 1* gezeigte Programm druckt genau eine Zeile, in diesem Fall mit 16 Zeichen, die im Adressenbereich 01D0...01DF gespeichert sind (von links nach rechts). Der notwendige Hardware-Aufwand ist minimal; im vorliegenden Fall wurden sechs Leitungen des I/O-Ports A für das ASCII-Zeichen und zwei Anschlüsse des Ports B für den Druckbefehl und die Zeichenanforderung verwendet (*Bild 2*). Das angegebene Programm ist beliebig verschiebbar, die angegebenen Adressen sind daher nur als relativ zu betrachten.

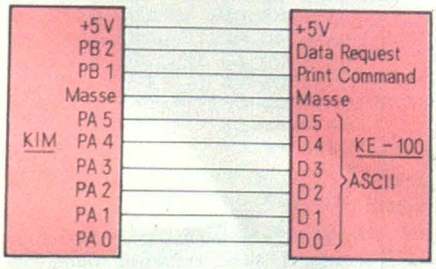


Bild 2. Verbindung des KE-100 mit den KIM-I/O-Ports

Da das vorliegende Programm die am Drucker ebenfalls vorhandene „Busy"-Leitung nicht abfragt, muß zwischen dem Ausdruck zweier Zeilen genügend Zeit „verschwendet" werden, um dem Motor des Druckers das Zurücklaufen zu gestatten. Diese Zeit ergibt sich allerdings automatisch, wenn man z. B. gleichzeitig das Auszudruckende auch noch auf einem Terminal mit serieller Schnittstelle und Geschwindigkeiten bis zu 1200 Bd ausgibt; andernfalls ist eine Verzögerungsschleife z. B. mit dem im KIM vorhandenen Timer vorzusehen. Fe.

Computer mit ROM-Kassetten

Der „Sorcerer", ein Personal Computer der amerikanischen Firma Exidy, ist nun auch in Deutschland von mehreren Firmen, z. B. vom Computershop Freiburg, erhältlich. Auffallend an dem etwa in der PET-2001-Preisklasse liegenden Gerät ist die Möglichkeit, fertige Programme seitlich (*Bild 1*) in das Gehäuse einzuschieben. Solch eine ROM-Kassette (*Bild 2*) kann bis zu 16 KByte an Programmen besitzen. Derzeit sind ROMs für Assembler, APL, PILOT, COBOL, FORTRAN und ein Disk-Operating-System (DOS) sowie ein Textverarbeitungs-Paket zu haben. Selbstverständlich ist auch eine direkte Programmierung in der Maschinensprache des Z-80-Prozessors möglich. Ein Standard-BASIC-Interpreter gehört zur Grundausstattung.

Bild 1. Die ROM-Kassetten beinhalten bis zu 16 KByte und werden seitlich in den „Sorcerer" eingeschoben

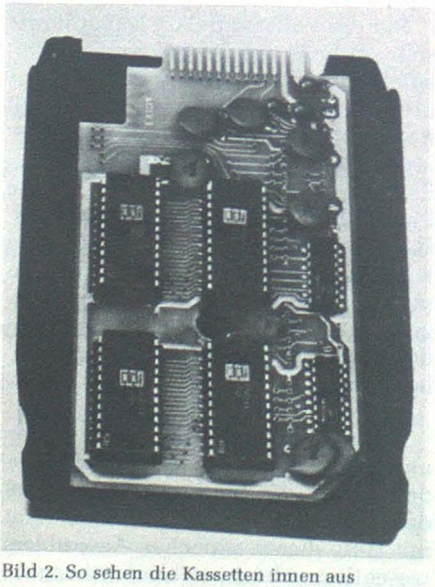


Bild 2. So sehen die Kassetten innen aus

Eine weitere Besonderheit ist die Darstellbarkeit hochauflösender Grafik auf dem Monitor-Bildschirm (512 x 240 Punkte). Der Zeichengenerator verfügt nicht nur über die üblichen ASCII-Symbole, sondern läßt sich außerdem noch frei programmieren. Dafür steht ein Satz von 128 Zeichen zur Verfügung.

Das eingebaute Kassetten-Interface ist von 300 Bd auf 1200 Bd umschaltbar. Für die Verbindung mit der übrigen Welt sorgen acht gebufferte I/O-Leitungen und ein serieller Anschluß für 300 Bd oder 1200 Bd. Fe.

Ing. (grad.)
Fred H. Heller

TI-59 berechnet Strahlungswiderstand

Das hier vorgestellte Programm für den Rechner TI-59 ermittelt den Strahlungswiderstand von Monopolen und Dipolen. Es stellt daher eine wertvolle Hilfe bei der Errechnung von Anpaßschaltungen für „unübliche" Antennenlängen dar.

Das Programm *(Bild)* arbeitet nach der Formel

$$R_s = 60 \cdot \int_0^{\pi/2} L^2(\varphi) \cos \varphi \, d\varphi$$

mit R_s = Strahlungswiderstand, L = vertikales Strahlungsdiagramm und φ = Elevationswinkel. Da der Rechner TI-59 leider nicht auf eine Integralsinus-Tabelle zurückgreifen kann, ist ein Unterprogramm vorhanden, das ihm die Werte für Integralsinus und -cosinus liefert; es belegt übrigens die Speicher 4, 7, 10, 11, 12, 13, 15, 16, 18 und die Programmarken A' und B'.

Für den Dipol gilt, daß sich der Strahlungswiderstand aus der Reihenschaltung zweier entsprechend halb so langer Monopole ergibt. Durch Druck auf die Label-Taste D werden zwei Flags gesetzt, die eine entsprechende Programmänderung bewerkstelligen; nach Abschluß der Dipolberechnung schalten sich die Flags automatisch wieder in die Monopol-Berechnungsart zurück.

```
000  91 R/S     041  19  19     082  55  ÷     123  65  ×     164  00  0     240  76 LBL
001  76 LBL     042  32 X:T     083  02  2     124  43 RCL     165  32 X:T     241  16 A'
002  11  A      043  08  8      084  95  =     125  22  22     166  77  GE     242  42 STO
003  42 STO     044  65  ×      085  23 LNX     126  54  )     167  00  00     243  10  10
004  14  14     045  89  π      086  42 STO     127  85  +     168  05  05     244  42 STO
005  91 R/S     046  95  =      087  25  25     128  01  1     169  01  1      245  15  15
006  76 LBL     047  22 INV     088  25 CLR     129  05  5     170  94 +/-     246  65  ×
007  12  B      048  77  GE     089  03  3     130  65  ×     171  34 fX      247  02  2
008  35 1/X     049  01  01     090  00  0     131  53  (     172  61 GTO     248  85  +
009  65  ×      050  69  69     091  65  ×     132  43 RCL     173  00  00     249  02  2
010  43 RCL     051  71 SBR     092  53  (     133  17  17     174  00  00     250  95  =
011  14  14     052  16 A'      093  00  0     134  39 COS     175  55  ÷     251  59 INT
012  95  =      053  42 STO     094  93  .     135  54  )     176  02  2     252  42 STO
013  61 GTO     054  20  20     095  05  5     136  65  ×     177  95  =     253  11  11
014  00  00     055  43 RCL     096  07  7     137  53  (     178  86 STF     254  01  1
015  26  26     056  19  19     097  07  7     138  00  0     179  02  02    255  42 STO
016  76 LBL     057  71 SBR     098  02  2     139  93  .     180  61 GTO     256  12  12
017  13  C      058  17 B'      099  01  1     140  05  5     181  00  00     257  42 STO
018  55  ÷      059  42 STO     100  06  6     141  07  7     182  30  30     258  13  13
019  03  3      060  21  21     101  85  +     142  07  7     183  65  ×     259  42 STO
020  00  0      061  43 RCL     102  43 RCL     143  02  2     184  02  2     260  04  04
021  00  0      062  17  17     103  24  24     144  01  1     185  95  =     261  42 STO
022  65  ×      063  71 SBR     104  75  -     145  06  6     186  32 X:T     262  16  16
023  43 RCL     064  16 A'      105  43 RCL     146  85  +     187  00  0     263  42 STO
024  14  14     065  42 STO     106  23  23     147  43 RCL     188  32 X:T     264  18  18
025  95  =      066  22  22     107  54  )     148  25  25     189  22 INV     265  61 GTO
026  87 IFF     067  43 RCL     108  85  +     149  85  +     190  77  GE     266  03  03
027  01  01     068  17  17     109  01  1     150  43 RCL     191  01  01     267  01  01
028  01  01     069  71 SBR     110  05  5     151  21  21     192  69  69     268  76 LBL
029  75  75     070  17 B'      111  65  ×     152  75  -     193  81 RST     269  17 B'
030  65  ×      071  42 STO     112  53  (     153  02  2     194  76 LBL     270  42 STO
031  04  4      072  23  23     113  43 RCL     154  65  ×     195  14  D     271  10  10
032  65  ×      073  25 CLR     114  17  17     155  43 RCL     196  86 STF     272  23 LNX
033  89  π      074  70 RAD     115  38 SIN     156  23  23     197  01  01     273  85  +
034  95  =      075  43 RCL     116  54  )     157  54  )     198  61 GTO     274  00  0
035  42 STO     076  17  17     117  65  ×     158  95  =     199  00  00     275  93  .
036  17  17     077  23 LNX     118  53  (     159  87 IFF     200  05  05     276  05  5
037  65  ×      078  42 STO     119  43 RCL     160  02  02
038  02  2      079  24  24     120  20  20     161  01  01
039  95  =      080  43 RCL     121  75  -     162  83  83
040  42 STO     081  17  17     122  02  2     163  32 X:T
```

(Fortsetzung des Programms auf Seite 266)

```
277  07  7     296  01  1     315  94  +/-   334  10  10    353  00  0     372  65  ×
278  07  7     297  42  STO   316  49  PRD   335  45  YX    354  42  STO   373  01  1
279  02  2     298  16  16    317  16  16    336  43  RCL   355  04  04    374  00  0
280  01  1     299  42  STO   318  69  OP    337  12  12    356  42  STO   375  00  0
281  06  6     300  18  18    319  27  27    338  55  ÷     357  07  07    376  00  0
282  95  =     301  43  RCL   320  69  OP    339  43  RCL   358  42  STO   377  00  0
283  42  STO   302  11  11    321  24  24    340  13  13    359  10  10    378  00  0
284  15  15    303  32  X:T   322  43  RCL   341  55  ÷     360  42  STO   379  95  =
285  43  RCL   304  43  RCL   323  04  04    342  43  RCL   361  11  11    380  59  INT
286  10  10    305  07  07    324  49  PRD   343  18  18    362  42  STO   381  55  ÷
287  65  ×     306  77  GE    325  18  18    344  65  ×     363  12  12    382  01  1
288  03  3     307  03  03    326  69  OP    345  43  RCL   364  42  STO   383  00  0
289  55  ÷     308  53  53    327  24  24    346  16  16    365  13  13    384  00  0
290  02  2     309  02  2     328  43  RCL   347  95  =     366  42  STO   385  00  0
291  85  +     310  44  SUM   329  04  04    348  44  SUM   367  16  16    386  00  0
292  03  3     311  12  12    330  49  PRD   349  15  15    368  42  STO   387  00  0
293  95  =     312  44  SUM   331  18  18    350  61  GTO   369  18  18    388  95  =
294  42  STO   313  13  13    332  25  CLR   351  03  03    370  43  RCL   389  92  RTN
295  11  11    314  01  1     333  43  RCL   352  01  01    371  15  15
```

Das TI-59-Programm. Die Programmschritte 201...239 sind hier freigelassen; bei 240 beginnt die Routine zur Errechnung der Integrale

Eingabe der Werte in den TI-59

Schritt	Prozedur	Taste	Anzeige
1	Antennenlänge eingeben	A	l
2	Nur bei Dipol: Flags setzen	D	l
3	Wellenlänge eingeben	B	R_s
3	oder Frequenz eingeben	C	R_s
4	Für andere Frequenzen bei der gleichen Ant. weiter bei Schritt 2		
	Berechnung des Integralsinus/-cosinus:		
5	X eingeben	A'	Si (X)
6	X eingeben	B'	Ci (X)

Die Eingabemöglichkeiten sind recht komfortabel. Will man z. B. für 27 MHz entsprechend 11 m Wellenlänge den Strahlungswiderstand eines 5,5 m langen Monopols errechnen, also für einen Halbwellen-Strahler, so kann man entweder 11 mit B oder 27 mit C eingeben, 5,5 dagegen mit A *(Tabelle)*.

Zur Berechnung des Strahlungswiderstandes bei diesem Beispiel (Ergebnis: 99,5 beim Monopol und 73,1 beim Dipol) braucht der Rechner rund fünf Minuten, was auf die recht langwierige Integralsinus- und -cosinus-Reihenentwicklung zurückzuführen ist. Für kleinere Antennenlängen im Verhältnis zur Wellenlänge braucht der TI-59 deutlich weniger Zeit, da die Schleifenzahl zur Anpassung an die erforderliche Genauigkeit automatisch geändert wird.

Literatur

Meinke/Gundlach: Taschenbuch der Hochfrequenztechnik, 3. Aufl., S. 526.

RAM — auf ROM umschaltbar

Eine RAM-Speicherzellentechnik mit bemerkenswerten Eigenschaften wurde von Hughes Microelectronics Ltd. geschaffen. Die Technik nennt sich NOVRAM und ist aus der CMOS-Technologie entwickelt worden. Betreibt man die Speicherzelle z. B. mit 5 V Versorgungsspannung, so verhält sie sich als normaler Schreib-Lese-Speicher, also RAM. Ein 10 ms langer 12-V-Impuls genügt aber bereits, um den Speicher in einen nichtflüchtigen Zustand zu versetzen, so daß er sich als ROM (Read-Only Memory) verhält. Das Prinzip beruht auf einem Tunneleffekt in dünnen Oxidschichten. Beim nächsten Schreibzyklus wird dann automatisch wieder der RAM-Zustand hergestellt. Ein besonderer Vorteil ist, daß die Zugriffszeiten der Zelle nicht höher als bei herkömmlichen RAM-Chips liegen.

Wann die Technik zur Serienreife gelangt, läßt sich noch nicht ganz absehen; Hughes stellte aber bereits einen Sechs-Dekaden-Zähler damit vor.

µC-Nachrichten

Eine **Dienstleistung besonderer Art** bietet die Schweizer Firma BD-Electronic in Basel an: Besitzer von Mikrocomputern wie TRS-80, PET 2001 oder Apple II können sich ihr Programm, das auf einer Kassette mitzubringen ist, auflisten lassen; dafür stehen zwei Drucker zur Verfügung. Die Gebühr dafür ist mit 0,50 Fr. für eine A-4-Seite recht bescheiden. Es besteht auch die Möglichkeit, Selbstklebeetiketten zu drucken, was gerade die Adressierarbeit für kleinere Firmen entlasten könnte.

□

Nach einer Untersuchung der Diebold GmbH wurden bis Mitte 1978 rund **7000 Hobbycomputer** oder „Personal Computers" abgesetzt, wovon ein großer Teil für Lehr- und Lernzwecke eingesetzt werde. Die Zahl 7000 scheint recht niedrig; die Untersuchung berücksichtigte allerdings weder Selbstbau-Systeme, noch privat aus den USA eingeführte Computer, noch Systeme, die von kleinen, unbekannten Firmen vertrieben werden. Daher gibt die Untersuchung als Mindestpreis für einen Mikrocomputer auch 650 DM an. Die tatsächlich existierende Zahl von kleinen Systemen dürfte deshalb erheblich höher liegen. Bereits für 1979 rechnet selbst Diebold mit einem Absatz von 15 000...20 000 Einheiten und einen Umsatz von 50...70 Mio. DM.

□

Die Firma Hofacker in Holzkirchen liefert verschiedene **Interface-Bausteine** für den PET 2001, so etwa einen Adapter auf eine RS-232-Schnittstelle, eine Digital/Analog-Wandler-Platine und eine Musikplatine zum gleichzeitigen Erzeugen von vier Tönen. Ferner werden vier unterschiedliche „Joysticks" für Spiele und ein I/O-Experimentier-Interface angeboten.

Hans Werner

Mikrocomputer

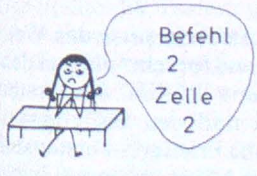
Eine kleine Einführung

12. Teil

So bearbeitet die CPU das Programm:

Die CPU holt sich den Inhalt der Speicherzelle 0, entschlüsselt und bearbeitet ihn.
Sie lädt also den Inhalt der Zelle 100 in den Akkumulator AC.

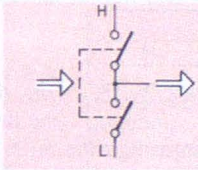
Die CPU holt sich jetzt den Inhalt der Speicherzelle 1 und führt die dort festgelegte Addition aus.
Das Ergebnis steht danach automatisch im Akkumulator.



Danach fragt sie den Inhalt der Speicherzelle 2 ab. In Zelle 2 steht: Speichere den Inhalt des Akkus in Zelle 1000. Damit sorgt sie dafür, daß die Summe in Zelle 1000 gelangt.

Der Datenbus

Rechenwerk, Programmzähler und Register arbeiten auf einen Datenbus, der über Ausgangstreiber (Puffer) rückwirkungsfrei nach außen geführt ist. Das Kontakt-Ersatzbild zeigt die Funktion für ein Bit. In Wirklichkeit sind acht elektronische Schalter vorhanden. Der Bus in der CPU steuert die

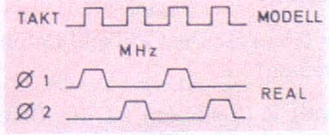


Schalter, deren Mittelpunkt auf den externen Datenbus führt. Die Puffer könen die Zustände H, L und HOCHOHMIG nach außen geben (Tri-State-Ausgänge).

beide	Schalter	offen	hochohmig
oberer	Schalter	geschlossen	H bzw. 1
unterer	Schalter	geschlossen	L bzw. 0

CPU-Steuereingänge

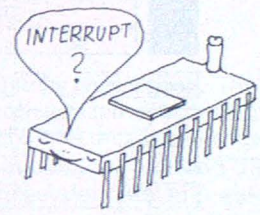
Die CPU hat 3 Steuereingänge:
Der Takt...
...synchronisiert alle Abläufe in der CPU. Aus 8 Impulsen bildet die Zentraleinheit einen Befehlszyklus. Das



Modell kommt mit einer einfachen Impulsreihe aus, während fast alle heutigen Prozessoren einen 2-Phasen-Takt benötigen. Die Frequenz wird in den meisten Fällen durch einen Oszillator erzeugt. Üblich sind 100 kHz bis mehrere MHz.

Reset
Ein Reset-Impuls (Rückstell-Impuls) unterbricht das Programm und stellt alle Register und Zähler auf Null. Es gibt kein eigentliches Startsignal. Der Grundzustand (Register und Zähler auf 0) wird durch Reset oder durch Einschalten der Stromversorgung erreicht. Der zuerst auszuführende Befehl muß in Speicherzelle 0 stehen.

Interrupt
(Prioritätsunterbrechung)
Mit diesem Eingang kann das Programm unterbrochen und an anderer Stelle – dem Interruptprogramm – fortgesetzt werden.



Am Schluß des Interruptprogramms kann ein Rücksprung zur alten Stelle erfolgen. Funktion und Bedeutung wird noch an Beispielen vertieft.

Der Steuerbus

Die Steuereingänge synchronisieren und steuern die Vorgänge in der CPU, die wiederum vier Steuersignale nach außen gibt. Von diesen Signalen sind alle angeschlossenen Bausteine abhängig. Die Leitungen werden als Steuerbus bezeichnet.

Der Steuerbus führt 4 Signale. Sie sind synchron zum Takt und steuern alle peripheren Bausteine, wie Speicher, Ein- und Ausgaberegister usw.; 8 Taktimpulse bilden einen Befehlszyklus. Danach wiederholen sich die Signale. Jeder Befehlszyklus ist in 4 Ausführungsphasen gegliedert:

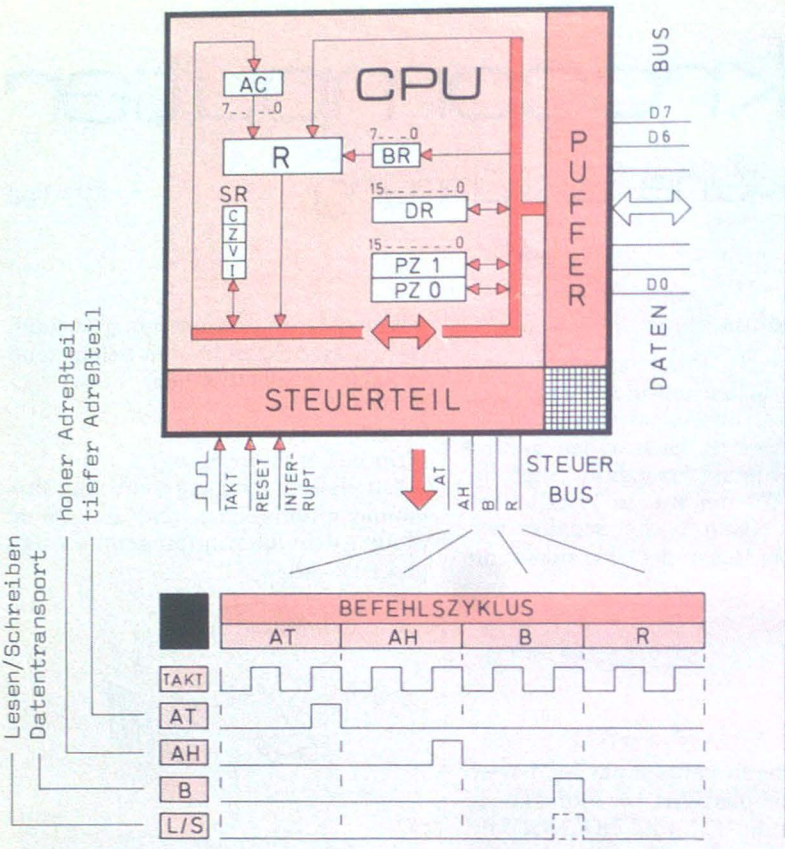
AT: erster Adreßteil

AH: zweiter Adreßteil an die Speicher

B: Daten oder Befehle holen bzw. ausgeben

R: Befehl ausführen

Wie wir später sehen werden, benötigt die vollständige Bearbeitung eines



Befehls mehr als einen Befehlszyklus. Mit AT und AH erscheint jeweils eine Hälfte der 16stelligen Adresse auf dem Datenbus.

Impuls AT:

Der tiefe Teil der Adresse (Stelle A7...A0) wird ausgegeben.

Impuls AH:

Der hohe Teil der Adresse (Stelle A15...A8) steht auf dem Datenbus.

Beispiel:

Der Programmzähler hat den Inhalt 00000011 00000001. Mit Impuls AT liegen die Pegel 00000001, mit AH die Pegel 00000011 an den Datenbusleitungen D7...D0.

Ausführungsphasen B und R

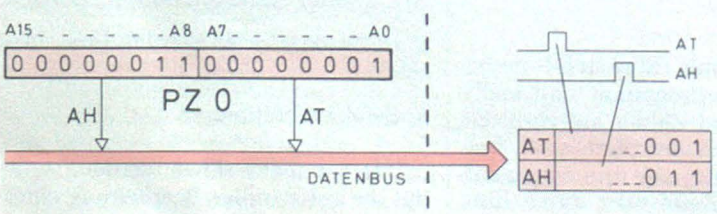
Alle externen Speicher werden mit den Impulsen B und L/S gesteuert. Während Intervall B können die Daten in die bzw. aus der CPU gelangen. Die Flußrichtung der Daten bestimmt Impuls L/S (Lesen/Schreiben). Im Gegensatz zu Impuls B ist L/S von dem gerade ausgeführten Befehl abhängig.

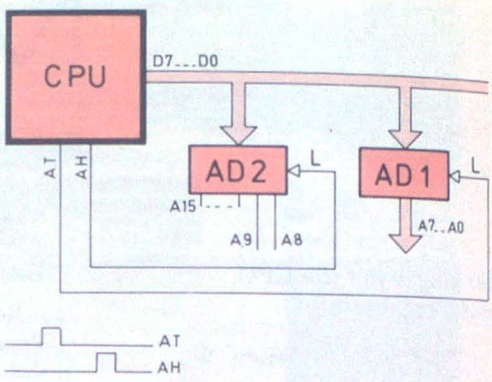
L/S = 1 bzw. H: Die CPU nimmt Daten auf (Lesen)

L/S = 0 bzw. L: Die CPU gibt Daten aus (Schreiben)

Intervall R (Rechnen) ist der Befehlsausführung in der CPU vorbehalten und hat für die äußere Steuerung keine Bedeutung.

Die 16stellige Adresse wird in 2 Hälften aus der CPU gegeben und muß extern gespeichert werden. Nur so sind nach Impuls AH alle Stellen gleichzeitig verfügbar. Das geschieht zweckmäßig durch die Adreßregister AD2 und AD1.





Sie werden mit den Ladeeingängen L gesteuert. Mit L = 1 übernehmen die Register neue Eingangsinformationen, die dann am Ausgang zur Verfügung stehen:

AT = 1:
Register AD1 wird mit dem niedrigen Teil A7...A0 der Adresse geladen.

AH = 1:
Register AD2 wird mit dem hohen Teil A15...A8 der Adresse geladen.

Fortsetzung folgt

µC-Nachrichten

Chuck Peddle, 6502-Designer und PET-2001-„Erfinder", kehrte Commodore den Rücken und machte sich selbständig. Dem 6502 bleibt er aber natürlich treu...

□

Der Postsonderpreis des Wettbewerbes **„Jugend forscht"** ging an den 18jährigen Hans Rüffler aus Freiburg im Breisgau und den 20jährigen Helmut Krüger aus Erlangen – ersterem für den Bau eines Mikrocomputer-Systems mit einem Taschenrechner-Chip für arithmetische Operationen und einem selbstgebauten Video-Terminal; letzterem für seine Arbeit „Messung des Maxwellschen Verschiebungsstromes". Übrigens wurde bereits 1967 der Postsonderpreis an den Entwickler einer programmierbaren Rechenanlage verliehen – sie war damals noch relaisgesteuert.

Axel Schnell

Interface von Mikrocomputern zu programmierbaren Taschenrechnern

Manch einer unserer Leser hat sicher schon darüber nachgedacht, wie man den recht verbreiteten Drucker PC 100 A, der u. a. zu dem Taschenrechner TI 59 paßt, zum Ausdruck von Mikrocomputer-Programmen verwenden kann. Das ist in der Tat nicht ganz unproblematisch und funktioniert nur auf eine recht unkonventionelle Weise.

Vorhanden war beim Autor ein programmierbarer Taschenrechner SR 52 der Firma Texas Instruments und der dazugehörende Drucker PC 100 A. Es sollte eine Möglichkeit gefunden werden, diesen Drucker extern anzusteuern, z. B. mit einem Mikrocomputer über einen 8-bit-Input-/Output-Port.

Eine direkte Ansteuerung über den 11poligen Stecker des Druckers war nicht möglich, da die Informationsübergabe vom Rechner zum Drucker seriell und in einer kaum zu durchschauenden Logik erfolgt.

Es blieb als einfachste Möglichkeit die Ansteuerung über das Tastenfeld des Rechners, welches zu diesem Zweck extern zugänglich gemacht wurde. Die neun Tastenreihen der Rechner von Texas Instruments werden der Reihe nach – durch einen kurzen 10-V-Impuls – abgefragt, und je nachdem, in welcher Spalte eine Taste gedrückt ist, wird der Reihenabfrageimpuls auf eine der fünf Spalten des Rechners durchgeschaltet. Aus der aktuellen Reihe und Spalte der Tastenmatrix erkennt der Rechner die gedrückte Taste und bildet den entsprechenden Tastencode. Beim TI 58/59 ist der Grundtastencode jeder Taste tatsächlich durch die entsprechenden Reihen- und Spalteninformation gegeben; beim SR 52 bilden die numerischen Tasten 0...9 eine Ausnahme, indem diese Tasten einer nicht vorhandenen Spalte 0 zugeordnet werden. Das Tastenfeld des SR 52 ist demnach in einer 6 x 10-Matrix organisiert (*Bild 1* und *Bild 2*), während das Tastenfeld des TI 58/59 aus einer 5 x 9-Matrix gebildet wird.

Eine einfache Möglichkeit, die Tastenfunktion extern nachzuvollziehen, ist durch zwei Analog-Multiplexer-/Demultiplexer gegeben. Die neun (zehn) Reihenabfrageimpulse werden direkt auf die Eingänge 1...9 (0...9) des 16fach-Analog-Demultiplexers 4067 gegeben und je nach anliegender 4-bit-Adresse auf den Ausgang Z durchgeschaltet. (Die in Klammern angegebenen Werte beziehen sich auf das Rechnermodell SR 52, während die ohne Klammer angegebenen Werte sich auf den TI 58 bzw. TI 59 beziehen.) Z wird auf den Eingang des 8fach-Analog-Multiplexers 4051 gegeben und dort je nach anliegender 3-bit-Adresse auf einen der Ausgänge 1...5 (0,1...5) durchgeschaltet (*Bild 4*). Die Ausgänge des 4051 sind mit den entsprechenden Spalten des Tastenfeldes verbunden. Die vier am 4067 anliegenden Bits bestimmen also beim TI 58/59 den Tastencode der ausgewählten Reihe, während die drei am 4051 liegenden Bits die ausgewählte Spalte bestimmen. Sind diese drei Bits beim SR 52 durch 000 gegeben, erfolgt mit den vier Bits der 4067-Adresse eine entsprechende numerische Eingabe in den Rechner. Nicht alle Bit-Kombinationen der 4 + 3-bit-Adresse werden ausgenutzt. Da die sechs (fünf) ungenutzten Eingänge des 4067 auf Masse gelegt werden, bewirkt eine ungültige Adreßkombination keinerlei Eingabe. Durch den Enable-Eingang der Analogschalter kann erreicht werden, daß die Nachbildung der Tastenfunktion nur

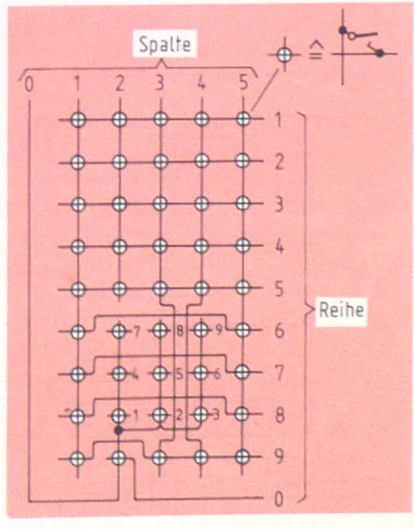


Bild 1. Tastaturmatrix des Rechners SR 52 von Texas Instruments

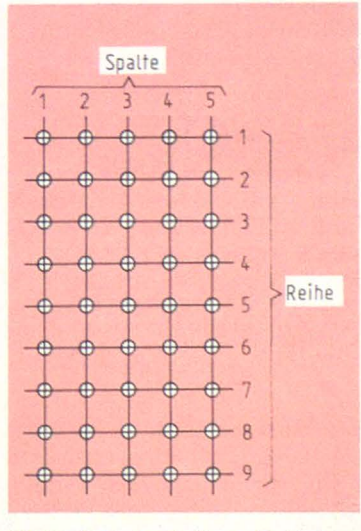


Bild 2. Tastenbeschaltung der Rechner TI 58 und TI 59

kurzzeitig erfolgt. Daher ist der Enable-Eingang des 4067 extra zugänglich gemacht und bildet mit den sieben Bits der Tastencodekombination ein 8-bit-Wort. Wenn dieses achte Enable-Bit nach jeder externen Tasteneingabe wieder hochgesetzt wird, kann der Taschenrechner zwischen den externen Eingaben ganz normal über das eigene Tastenfeld bedient werden.

Bei den verwendeten Analogschaltern muß unbedingt darauf geachtet werden, daß die Spannungen der Eingangs- und Ausgangspins in keinem Fall die Versorgungsspannung der Bausteine über- bzw. unterschreiten. Daher ist es nicht möglich, die beiden Bausteine durch die Batterie-Spannung (4,2 V) des Rechners zu versorgen, da die Abfrageimpulse eine Amplitude von etwa 10 V haben. Diese Spannung wird intern im Rechner als Versorgungsspannung der Rechnerchips von einem Spannungswandler erzeugt. Man kann nun entweder intern diese 10-V-Versorgungsspannung anzapfen (der Stromverbrauch der Analogschalter liegt weit unter 1 mA) oder eine potentialmäßig passende externe Spannungquelle mit einer Spannung von 10...15 V verwenden.

Die sieben Adreßbits der beiden Analogschalter und der Enable-Eingang ergeben insgesamt acht Bits, die von einem CMOS-kompatiblen 8-bit-Datenbus oder Input-/Output-Port eines Mikroprozessors oder Hobbycomputers direkt angesteuert werden können. Üblicher ist jedoch eine TTL-kompatible

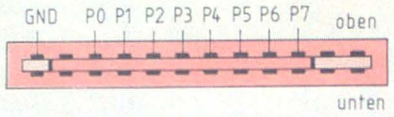


Bild 5. Anschlußbelegung des PET-User-Port, von hinten gesehen

Ansteuerung mit 5 V. Dazu wurde vor die sieben Adreßeingänge und den Enableeingang noch ein einfaches TTL-CMOS-Interface geschaltet (Bild 3), wobei jetzt zu beachten ist, daß die Eingangsinformation invertiert sein muß.

Mit der beschriebenen Schaltung ergibt sich eine einfache Möglichkeit, den weitverbreiteten Thermodrucker PC 100 als Ausgabemedium eines Mikrocomputers zu verwenden und gleichzeitig noch die starke mathematische Potenz der programmierbaren Taschenrechner zu nutzen. Die Ausgabeinformation des Mikrocomputers muß allerdings in diesem erst in einem kleinen Basic- oder Maschinenprogramm in das entsprechende Eingabeformat des verwendeten Taschenrechners umgewandelt werden. Die Programmierbarkeit des Taschenrechners kann insofern ausgenutzt werden, als einfache Umrechnungen bzw. der nicht durch einen einzigen Tastendruck zugängliche Print-Befehl auf jeweils eine der vom Benutzer programmierbaren Tasten A...E gelegt werden kann.

Die maximale Betätigungsfrequenz der Tasten liegt für die einfache Zifferneingabe bei etwa 50 Hz. Sobald interne Umrechnungen im Taschenrechner erfolgen, sinkt die maximale Eingabefrequenz jedoch stark ab. Hierauf ist bei der Ausgabe durch den externen Mikrocomputer zu achten.

Die beschriebene Schaltung ist nicht nur für die programmierbaren Rechner von Texas Instruments anwendbar, sondern für alle Rechner, die das Tastenfeld in gleichartiger Weise abfragen. Dies ist bei den meisten heute erhältlichen Rechnern der Fall.

PET-Ausgabeprogramm für den SR 52

```
05  REM USERPORT AUF OUTPUT
    SCHALTEN
10  POKE 59471, 0 : POKE 59459, 255
15  INPUT B
20  B$ = STR $ (B) : C = LRN (B$7 +1
25  FOR I = 1 TOC : V$ = MID $ (B$, I, 1)
30  IF V$ = "." THEN A = 198 : GOTO 70
35  IF V$ = "–" THEN A = 182 : GOTO 70
40  IV V$ = "E" THEN A = 218 : GOTO 70
45  IF I<>C THEN A = 255 : GOTO 65
50  A = 237 : REM CODE FOR "2ND"
55  GOSUB 85
60  A = 198 : GOTO 70 : REM CODE FOR
    "."
65  V = VAL (V$) : A = 255–V
70  GOSUB 85
75  NEXT I
80  GOTO 15
82  REM UNTERPROG. A AN USERPORT
85  POKE 59471, A
86  FOR I = 1 TO 5
87  NEXT
90  POKE 59471,0
95  RETURN
```

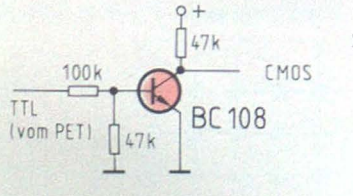


Bild 3. Schaltung zur TTL/CMOS-Pegelanpassung. Sie wird inzwischen den PET-Anschlüssen P 0...7 und den Anschlüssen E sowie A 0...6 benötigt

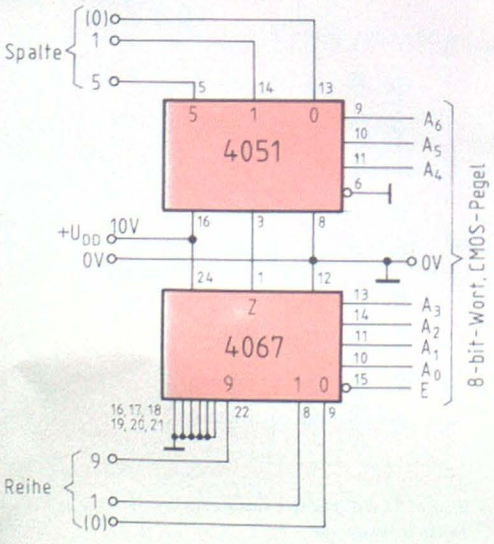


Bild 4. ▶ Interfaceschaltung zur externen Ansteuerung des Tastenfeldes

Ratschläge für Computer-Neulinge

Während der Lieferzeit...

...Ihres Mikrocomputers sollten Sie die Zeit nutzen, sich ein passendes Netzgerät zu besorgen oder selbst zu bauen. Vermeiden Sie unbedingt die Verwendung mehrerer, unglücklicherweise meist auch zu höheren Spannungswerten hin regelbarer Netzgeräte, da sonst nie sichergestellt werden kann, daß im richtigen Moment die richtige Spannung vorhanden ist. Ein spezielles Mikrocomputer-Netzteil ist billiger als eine neue Mikrocomputer-Platine!

Terminals mit serieller ASCII-Schnittstelle...

...sollten entsprechend der Mikrocomputer-Schnittstelle programmiert

werden. In den meisten Fällen bedeutet dies: Kein Parity-Bit, acht Datenbits, bei 110 Bd zwei Stopbits, darüber 1 Stopbit. Das achte Bit legt man zweckmäßig auf L-Pegel (null), das geschieht am UART-Anschluß D_7.

Wenn der Computer schweigt

Sollte auf dem Terminal-Bildschirm außer den eingetippten Zeichen nichts erscheinen, so programmieren Sie den Mikrocomputer für eine einfache Testschleife, die irgendein ASCII-Zeichen pausenlos ausgibt, z. B. den Buchstaben A. Beim KIM-1 kann dieses Programm etwa so aussehen:

A9 00 8D F3 17 A9 74 8D
F2 17 A9 41 20 A0 1E 18
90 F8

Schreiben Sie dieses Programm irgendwo in Ihren Speicher, z. B. ab der Adresse 0000 (dies kann mit dem KIM-Display und der Tastatur auf der KIM-Platine geschehen) und starten Sie es. Es druckt pausenlos den Buchstaben A mit einer Geschwindigkeit von 600 Bd aus, was die Fehlersuche z. B. in den Interface-Schaltungen sehr erleichtert.

Endlich läuft das System

Ist endlich die Kommunikation mit dem Computer möglich, so vergessen Sie nicht, auch einmal einen Blick in das Systemhandbuch zu werfen. (Software- und vor allem Hardware-Handbücher sind im Augenblick noch nicht wichtig.) Versuchen Sie, auch wenn Ihnen zunächst alles spanisch vorkommt, einfach zwei Adresseninhalte zu addieren und an eine dritte Adresse abzuspeichern; das KIM-Handbuch gibt dafür ein einfaches Beispiel. Nach diesem ersten Programm sieht alles schon ganz anders aus! *Fe.*

Karl Schedler

„Knossos" – ein Interpreter für den Nascom-1

Hinter dem Namen Knossos verbirgt sich ein von Gerhard Bundle entwickelter Texteditor und Interpreter besonderer Art, der vom ABC-Computershop in München für den Mikrocomputer Nascom-1 vertrieben wird. Knossos wird auf einem 1-K-EPROM (2708) geliefert, das lediglich in den freien Erweiterungssockel des Nascom eingesteckt zu werden braucht. Mit dem Interpreter lassen sich u.a. effektvolle Spielprogramme (mit Zufallsgenerator und Arithmetik), überhaupt Programme, die umfangreiche Textausgaben erfordern, wesentlich einfacher und schneller schreiben, als das bei Maschinenprogrammierung der Fall wäre. Er hat eine eigene, schnell erlernbare, konsequent und logisch aufgebaute Syntax. Der Texteditor ermöglicht dabei ein bequemes Eingeben und Korrigieren.

Es wird auch hier, wie allgemein üblich, unterschieden zwischen (Betriebs-)Kommandos (*Tabelle 1*), die im wesentlichen das Eintragen, Einfügen, Löschen und Anzeigen der Programmzeilen bewerkstelligen, und den (Programm-)Anweisungen (*Tabelle 2*), die vom Interpreter verstanden werden. Man kann also, wie man das von Texteditoren gewohnt ist, beliebig Zeilen einfügen oder löschen. Programmkorrekturen werden dadurch wesentlich erleichtert. Zeilennummern gibt es allerdings nicht, man muß sequentiell eintragen bzw. korrigieren.

Die Zeilen werden, so wie sie ins Video-RAM eingetragen wurden, der Reihe nach (1 Zeichen = 1 Byte) ins User-RAM geladen, wobei jede Zeile mit „CR" und jede Anweisung mit einem nicht in der Syntax vorkommenden Zeichen getrennt werden. Nach dem Starten beginnt dann der Interpreter der Reihe nach durch Vergleich die Anweisungen zu erkennen (sonst Fehlermeldung) und immer nach gleichem Schema abzuarbeiten. Hinter jeder Anweisung steht ein Argument, das grundsätzlich entweder aus einer Konstanten (Hex. od. Dez.), aus einer Variablen, einem Operator oder aus einem beliebigen Ausdruck, der diese Faktoren enthält, bestehen kann.

Der Wert des Ausdrucks, der nach üblichen Prioritätsregeln ermittelt wird, ist maßgebend. Intern wird übrigens immer hexadezimal (16 bit binär) gearbeitet, die Anzeige erfolgt immer dezimal. Praktischerweise werden auch Eingaben so behandelt. Jedem der 26 Variablennamen entspricht ein fester Platz in einem Teil des RAM-Bereichs, der sonst vom NASBUG belegt wird. So ist z.B. die Cursor-Adresse identisch mit der Variablen „H", die man per Programm beliebig belegen und damit komfortable Textausgaben erzeugen kann. Sehr nützlich auch der wie eine 8-bit-Variable zu behandelnde Byte-Operator, der ein beliebiges Byte im RAM repräsentiert.

Der Hex-Wert des Arguments nach dem Operatorzeichen entspricht der physikalischen Adresse. Die Feldvariablen werden ab dem vom Texteditor belegten RAM-Bereich aufwärts abgelegt. Die Feldgröße wird also nur vom vorhandenen Platz begrenzt. Durch geschicktes Programmieren lassen sich sogar Matrizen erstellen.

Trotz zahlreicher arithmetischer Programmbeispiele in der mitgelieferten Anleitung, kann man nicht ganz verbergen, daß für anspruchsvolles Rechnen die Möglichkeiten nicht ausreichend sind – nicht zuletzt deshalb, weil sich ohne umständliche Programmier-

Tabelle 1.
Knossos-Betriebskommandos

Zeichen	Bedeutung
N	muß bei jeder neuen Programmeingabe gedrückt werden.
(BS)	Einfügen oder Eintragen einer Programmzeile
K	Löschen einer Zeile
f	nächste Textzeile ausgeben
a	Zeilenzeiger an Anfang
-	Zeilenzeiger um eine Zeile zurück
D	File auf Kassette ausgeben
E	Programm starten

verrenkungen nur ganze Zahlen und, wegen der 16-bit-2er-Komplement-Darstellung, Werte bis max. ±32768 damit verarbeiten lassen. Aber wozu gibt es billige Taschenrechner?

Programmsprünge sind sehr einfach möglich, nämlich durch Setzen einer Marke vor Anweisungen. Für bedingte Verzweigungen gibt es drei Arten von Bedingungs-Marken (P, N, Z entsprechend Positiv, Negativ, Zero), die mit dem Inhalt eines Flag-Bytes verglichen werden; bei Übereinstimmung werden die nachfolgenden Anweisungen einer Zeile ausgeführt, andernfalls wird die Zeile übersprungen. Da es nur diese Marke – also keine unterscheidbaren Labels oder Zeilennummern – gibt, kann nur relativ gesprungen werden (vor und zurück). Das ist immer dann von Nachteil, wenn nachträglich noch Marken eingefügt werden müssen. Das Flag wird gesetzt durch die „Condition-Anweisung" je nach dem Wert ihres Arguments. Man muß allerdings beachten, daß dabei der Wert von Variablen verändert werden kann.

Die, in sämtlichen höheren Programmsprachen vorhandene, Laufanweisung kommt hier nicht vor, kann aber leicht programmiert werden. Selbstverständlich gibt es auch Unterprogramme, die mit einer Marke beginnen und mit R(eturn) enden, also auch relativ angesprungen werden.

Versierte Maschinensprache-Programmierer wird es vielleicht interessieren, die Bequemlichkeit von Knossos mit den Möglichkeiten des Maschinenprogramms zu verbinden. Auch das ist durch einen U(ser)-Programmaufruf möglich. Das Argument dahinter bedeutet, wie beim Byte-Operator, die Adresse. Was bei Aufrufen im Monitor problemlos abläuft, kann allerdings bei eigenen Programmen irgendwo im RAM zu bösen Überraschungen führen. Man muß beim Eingeben über Texteditor mitzählen und dann aufpassen, daß nachträgliche Einfügungen oder Feldvariable nicht plötzlich alles überschreiben. Die User-Programme sollten mit RET (C9) abgeschlossen sein, der Interpreter macht dann mit der nächsten Anweisung weiter. Eine Wertübergabe erfolgt dabei automatisch über Akku und dem niederwertigen Teil der Variable „A".

Wie man feststellen kann, kommen in Knossos in einfacher Form alle grundlegenden Strukturelemente, die auch höhere Programmsprachen aufweisen, vor. Deshalb dürfte es gerade für den Anfänger eine Möglichkeit sein, sich auf bequeme und schnelle Art in die Programmiermethodik überhaupt einzuarbeiten.

Tabelle 2. Knossos-Syntaxelemente

Zeichen	Bezeichng.	Beispiel	Erläuterung
		.C = A–10	Wertzuweisung
		$(A + B) = % F0	Byte-Zuweisung
C	Condition	C A–% 53	entsprechend dem Wert des Ausdrucks wird das Flag belegt.
J	Jump	J –2	Springe um 2 Marken zurück
		J (A + B)	...um den ganzzahligen Wert des Ausdrucks
S	Subroutine	S 4	wie bei Jump
R	Return	R	
U	User	U % 244	das Programm auf Adresse 0244 wird angesprungen.
?	Accept	? A	Eine Eingabe wird erwartet, der Wert
		?@(I + 1)	wird der Variablen oder Feld zugew.
T	Type	T6 A/100	Das Ergebnis A/100 wird 6 Zeichenpositionen vom Cursor rechtsbündig ausgegeben.
W	Write	WKNOSSOS	Der Text „KNOSSOS" wird ab Cursor ausg.
/	CRLF		bei der Ausgabe wird eine neue Zeile begonnen
>	Label	>. B = 5	Die Anweisung wird als Sprungziel markiert.
N, P, Z	Flags	Z J–6	Ist das Flag Z(ero) (letzte Condition-Anweisung), so wird die Sprunganweisung dahinter ausgeführt, sonst wird sie übersprungen.
E	End		

–100 (max. ±32767)	Dezimalkonstante
% EA4 (max. FFFF)	Hexkonstante
£	Zufallszahl (max. 32768)
A...Z	Variablen (16 bit)
$ (‹Ausdruck›)	Byteoperator (8 bit)
@(‹Ausdruck›)	Feldvariable (16 bit)
+ – / * ( )	vier Grundrechenarten mit Klammeroperationen
,	Module (Divisionsrest)

Mikrocomputer-Tastatur

Die Fortschritte der Integrationstechnik ermöglichen es, komplette Mikroprozessorsysteme einschließlich Stromversorgung, Speicher und Peripheriebausteinen in dem Gehäuse der Tastatur unterzubringen (*Bild*).

Die Tastatur in der Teletype-Anordnung ASR 33 ist mit dem Tastatur-System Rafi RS 76 C aufgebaut; andere Tastenanordnungen im Raster 5 x 18 Tasten sind möglich. Als Anzeige dient ein 20stelliges 16-Segment-Fluoreszensdisplay mit 9 mm hohen, grünen Ziffern. Ein Video- bzw. VHF-Anschluß ist als Option lieferbar, wobei auf dem Bildschirm 16 Zeilen mit jeweils 64 Zeichen dargestellt werden können. Für das System 8085 ist ein Speicher mit 12 KByte vorgesehen; 8 KByte dienen als Programmspeicher und 4 KByte

Neben der Tastatur und der Anzeige ist in dem Gehäuse ein komplettes 8085-Mikroprozessorsystem untergebracht

als Datenspeicher. Als Ein- und Ausgänge stehen aktive und passive serielle Schnittstellen sowie weitere 88 Parallel-Ein/Ausgänge zur Verfügung. Die seriellen potentialfreien Ein- und Ausgänge sind mit Stromschleife 20 mA oder 24 V lieferbar. Die Baudraten bis 9600 Baud sind per Programm einstellbar.

Das Gerät ist durch seine Flexibilität für viele Bereiche der Daten-, Regel-, Steuer- und Meßtechnik einsetzbar, da feste Funktionen erst durch die Programmierung festgelegt werden. Durch den Wegfall einer speziellen Hardwareentwicklung für den Rechnerteil können mit diesem Standardgerät Projekte kostengünstig und kurzfristig durchgeführt werden.

µP-Portrait:

MC 6800

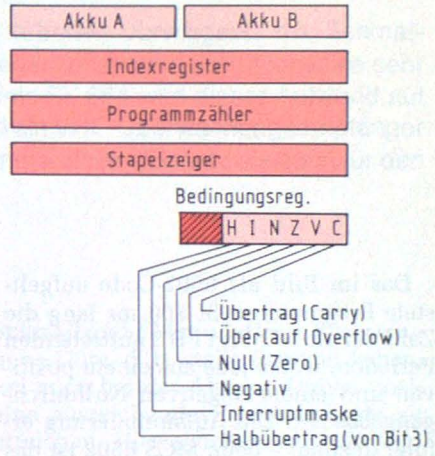
Der MC 6800 wurde 1974 von Motorola eingeführt und war damals der erste ernstzunehmende Konkurrent des 8080. Obwohl er mittlerweile zu den betagten 8-bit-Mikroprozessoren gehört, ist er keineswegs veraltet. Viele seiner Vorzüge gelten auch heute noch als modern. So hat sich beispielsweise die Versorgungsspannung von + 5 V inzwischen auch bei anderen Typen weitgehend durchgesetzt. Ungewöhnlich war zum damaligen Zeitpunkt auch die Tatsache, daß man keine gesonderten Ein/Ausgabe-Befehle mehr benötigt, sondern alle Peripherie-Elemente über Adressen anspricht (memory-mapped I/O).

Der Erfolg des MC 6800 ist zum großen Teil auf seinen einfachen und übersichtlichen Aufbau zurückzuführen. Das *Bild* zeigt die interne Struktur, wie sie sich für den Programmierer darstellt: Neben dem Programmzähler sind zwei weitere 16-bit-Register vorhanden, das Indexregister und der Stapelzeiger (*Stackpointer*). Mit Hilfe des Stapelzeigers kann man den Stapelspeicher (*Stack*) auf jeden beliebigen Bereich innerhalb des gesamten Adreßraumes von 64 KByte legen – andere Prozessoren stellen oft nur eine begrenzte Anzahl von Speicherzellen für den Stack zur Verfügung, dadurch wird natürlich die Zahl der Unterprogrammsprünge ebenfalls eingeschränkt. Das Indexregister kann wie ein Zeiger auf beliebige Adressen deuten; mit Hilfe der indizierten Adressierung ist dann ein Zugriff zu den nächsten 255 Bytes möglich.

Eine Besonderheit stellen die beiden Akkumulatoren A und B dar. Bis auf wenige Ausnahmen führen beide die gleichen Operationen aus. Mit den Befehlen ADC (*add with carry*), SBC (*subtract with carry*) und verschiedenen Schiebebefehlen sind sehr leicht effektive 16-bit-Operationen zu realisieren.

Im Bedingungsregister sind lediglich sechs Bits relevant, sie dienen hauptsächlich für bedingte Sprünge, die alle in der relativen Adressierungsart ausgeführt werden. Insgesamt sind 16 verschiedene relative Sprungbefehle möglich. Die Versatzadresse besteht bei ihnen aus einem Byte, das als Dualzahl mit Vorzeichen interpretiert wird.

Die 72 Grundbefehle des MC 6800 lassen sich zum größten Teil in mehreren Adressierungsarten ausführen. Eine kurze Erklärung dieser Adressierungsarten findet sich in der *Tabelle*.



Registerstruktur des 6800

Wie daraus zu ersehen ist, hat der Prozessor 1-, 2- und 3-Byte-Befehle. Die Dauer hängt in erster Linie von der Adressierungsart ab, sie kann 2...12 Zyklen betragen. Der längste Befehl ist die SWI-Instruktion, die einen Software-Interrupt auslöst und sehr nützlich für Testzwecke ist. Neben dem Reset sind noch zwei weitere Interrupts vorhanden: ein nicht maskierbarer und ein maskierbarer, der nur ausgeführt wird, wenn das Interruptmaskenbit im Bedingungsregister nicht gesetzt ist. Die Startadressen für alle Interrupt-Service-Routinen müssen an den obersten Speicherplätzen hinterlegt werden.

Der MC 6800 wird gegenwärtig auch noch von den Firmen AMI, Thomson-CSF, Hitachi und Fairchild angeboten. Von Motorola ist er für die Taktfrequenzen 1 MHz, 1,5 MHz und 2 MHz und für die Temperaturbereiche 0...70, –40...85 und –55...125 °C lieferbar. Das Angebot an Peripherie-Bausteinen für diese CPU gehört zu den umfangreichsten auf dem Markt. Erst kürzlich wurden vorgestellt: Advanced-Data-Link-Controller, DMA-Controller, CRT-Controller, Floppy-Disk-Controller und Dual-Floppy-Disk-Controller.

Als Einchip-Mikrocomputer wird der 6800 inzwischen mit ROM, RAM, Zeitgeber, Taktgeber und E/A-Einheit unter der Bezeichnung 6801 angeboten.

Rudolf Hofer

Tabelle: Adressierungsarten des 6800

Adressierung	Bemerkung
Akkumulator-A. und Inherent	Im Operationscode steckt bereits die Information, wo der Operand zu finden ist, z. B. ABA (addiere Akkus B und A).
Unmittelbar (immediate)	Der Befehl enthält den Operanden selbst, z. B. LDAA 10 (lade Akku A mit 10).
Direkt	Das zweite Befehls-Byte gibt die Adresse des Operanden an, und zwar in der Reihenfolge höherwertiges Byte – niederwertiges Byte.
Indiziert	Die Adresse ergibt sich aus der Summe des zweiten Befehls-Bytes und dem Inhalt des Indexregisters.
Erweitert (extended)	Das zweite und dritte Byte geben die Operanden-Adresse an.
Relativ	Die Adresse ergibt sich aus dem gegenwärtigen Programmzählerstand und dem zweiten Befehls-Byte (Sprünge sind von –128... +127 möglich).

Frequenzmessung per Software

KIM als Nf-Zähler

Sowohl in der Analogtechnik als auch in der Mikrocomputer-Technik müssen manchmal Frequenzen gemessen werden. Was liegt näher, als den Mikrocomputer selbst damit zu beauftragen? So lassen sich z. B. Modem-Tonfrequenzen oder Kassetten-Interfaceschaltungen exakt abgleichen, ohne einen teuren Frequenzzähler zu benötigen.

Das im *Bild* als 6502-Code aufgelistete Programm mißt 100 ms lang die Zahl der am I/O-Port PB 3 auftretenden Perioden, wobei jede aus einem positiven und einem negativen Nulldurchgang besteht. Die Aufsummierung erfolgt dezimal – beim MCS 6502 ist das besonders einfach, da sich dieser Mikroprozessor mit dem Befehl SED (Set Decimal) in die dezimale Arithmetik umschalten läßt.

Nach diesen 100 ms tritt ein Interrupt des programmierbaren Timers auf. Das Programm springt an die Adresse 003B, setzt den Stackpointer auf seine Grundstellung zurück und zeigt etwa 260 ms lang das Zählergebnis an. Da die Auflösung 10 Hz beträgt, muß dafür gesorgt werden, daß – um eine Anzeige in Hz zu erreichen – noch genau eine Stelle folgt; d. h. die niederwertigste Stelle des Zählergebnisses muß in die vorletzte Anzeigenstelle geschoben werden. Dies erfolgt mit LSR- und ROR-Befehlen; zu beachten ist, daß die 6502-Chips, die vor dem Juni 1976 gefertigt wurden, den ROR-Befehl nicht kennen! In diesem Fall müßte das Programm geringfügig umgeschrieben werden, um die entsprechende Befehlsfolge durch eine andere, die kein ROR beinhaltet, zu ersetzen.

```
0000 LDA #$FF
0002 STA 1701
0005 LDA #$00
0007 STA   FA
0009 STA   FB
000B STA 17FB
000E LDA #$80
0010 STA 1700
0013 LDA #$3B
0015 STA 17FA
0018 LDA #$62
001A STA 170F
001D LDA #$08
001F BIT 1702
0022 BNE 001F
0024 BIT 1702
0027 BEQ 0024
0029 SED
002A LDA   FA
002C SEC
002D ADC #$00
002F STA   FA
0031 LDA   FB
0033 ADC #$00
0035 STA   FB
```

```
0037 CLD
0038 JMP 001D
003B LDX #$FF
003D TXS
003E CLD
003F STX 1707
0042 LDA #$00
0044 STA   F9
0046 LDX #$04
0048 LSR   FB
004A ROR   FA
004C ROR   F9
004E DEX
004F BNE 0048
0051 JSR 1F1F
0054 BIT 1707
0057 BPL 0051
0059 JSR 1F6A
005C CMP #$10
005E BPL 0066
0060 ASL   A
0061 ASL   A
0062 ASL   A
0063 ASL   A
0064 STA   0F
0066 JMP 0000
```

```
0000 A9 FF 8D 01
0004 17 A9 00 85
0008 FA 85 FB 8D
000C FB 17 A9 80
0010 8D 00 17 A9
0014 3B 8D FA 17
0018 A9 62 8D 0F
001C 17 A9 08 2C
0020 02 17 D0 FB
0024 2C 02 17 F0
0028 FB F8 A5 FA
002C 38 69 00 85
0030 FA A5 FB 69
0034 00 85 FB D8
0038 4C 1D 00 A2
003C FF 9A D8 8E
0040 07 17 A9 00
0044 85 F9 A2 04
0048 46 FB 66 FA
004C 66 F9 CA D0
0050 F7 20 1F 1F
0054 2C 07 17 10
0058 F8 20 6A 1F
005C C9 10 10 06
0060 0A 0A 0A 0A
0064 85 0F 4C 00
0068 00 00 00 00
```

Dieses kurze Programm macht aus dem Mikrocomputer KIM-1 einen Nf-Zähler mit 10 Hz Auflösung. Die Startadresse ist 0000

Das zu messende Eingangssignal kann auf zwei Arten dem Mikrocomputer zugeführt werden; entweder legt man es als TTL-Pegel direkt an den I/O-Port PB 3, oder man verwendet eine Schaltung, die bereits unter dem Titel „Oszilloskop + KIM = Speicheroszilloskop" in Heft 22/1978 vorgestellt wurde. Dann ist nämlich über die KIM-Tastatur eine digitale Einstellung des Triggerpegels möglich, wobei die Taste O dem Pegel –2,55 V und die Taste F dem Pegel +2,4 V entspricht. Selbstverständlich sind alle Zwischenwerte in Schritten von etwa 0,3 V einstellbar, und die Taste 8 legt den Triggerpegel auf 0 V.

Um zu erreichen, daß der Timer-Interrupt einen nicht maskierbaren Interrupt (NMI) auslöst, ist es notwendig, eine Verbindung zwischen PB 7 und NMI herzustellen. Der NMI-Vektor (003B) wird automatisch vom Programm in die Zellen 17FA und 17FB gespeichert. Außer den Display-Registern 00F9, 00FA, 00FB sowie dem Programm selbst werden keine Speicherplätze belegt. Die Bedeutung der Unterprogramm-Sprünge in das KIM-Monitorprogramm ist dem im Franzis-Verlag erschienenen Sonderheft „Hobbycomputer" zu entnehmen.

Herwig Feichtinger

Hans Werner

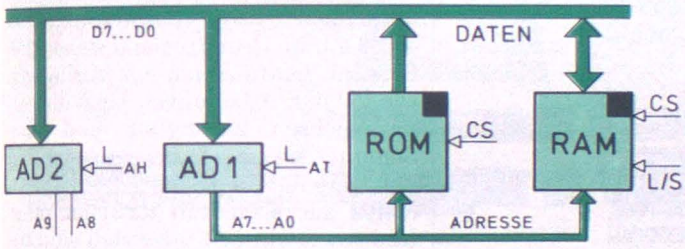
Mikrocomputer

Eine kleine Einführung

13. Teil

Speicher

Wir wollen an die CPU 2 Speicher anschließen. Beide Speicher haben eine Kapazität von 256 Byte, kommen also mit 8 Adreßeingängen aus ($2^8 = 256$).



Es genügt eine Hälfte der 16stelligen Adresse. Die anderen 8 bit werden zur Selektierung (Auswahl) der Speicher und später zur Steuerung der Ein- und Ausgaberegister herangezogen. Auf die genaue Bedeutung der Begriffe ROM und RAM soll vorläufig verzichtet werden. Wir lassen unser Strichmännchen träumen und wenden uns den prinzipiellen Speicherunterschieden zu.

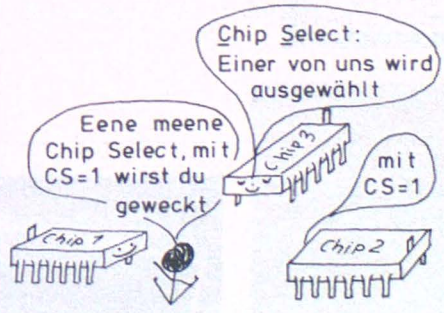
Das ROM
ist der Programmspeicher. Aus ihm können nur Daten ausgelesen werden. Das ist vernünftig, denn Mikrocomputer sind als fertig programmierte Einheit in die entsprechenden Geräte eingebaut und arbeiten immer mit demselben Programm. Der Speicherinhalt des ROM ist unabhängig von der Stromversorgung und kann nicht verloren gehen.

Das RAM
dient als Kurzzeitspeicher für Zwischenergebnisse. Es können Daten ein- und ausgegeben werden. Das ROM besitzt nur den Steuereingang CS (Chip Select), das RAM noch zusätzlich L/S.

Der Steuereingang CS
CS = 0:

Der Datenausgang der Speicher ist hochohmig vom Datenbus getrennt. Es ist kein Datentransport möglich.

CS = 1:
Es können Daten ein- bzw. ausgegeben werden.
Mit dem Steuereingang CS kann also der gewünschte Speicher ausgewählt und aktiviert werden.



Der Steuereingang L/S
L/S = 0: Schreiben
Es können Daten in das RAM eingegeben und gespeichert werden.
L/S = 1: Lesen
Es können Daten aus dem Speicher gelesen werden (Ausgabe).
Natürlich muß in beiden Fällen CS = ...sein, weil sonst...

CS = 1 muß sein, weil sonst der Speicher hochohmig vom Datenbus getrennt ist.

Zusammenfassung

CS	ROM
0	hochohmig getrennt
1	Lesen

CS	L/S	RAM
0	0	hochohmig
	1	getrennt
1	0	Schreiben
	1	Lesen

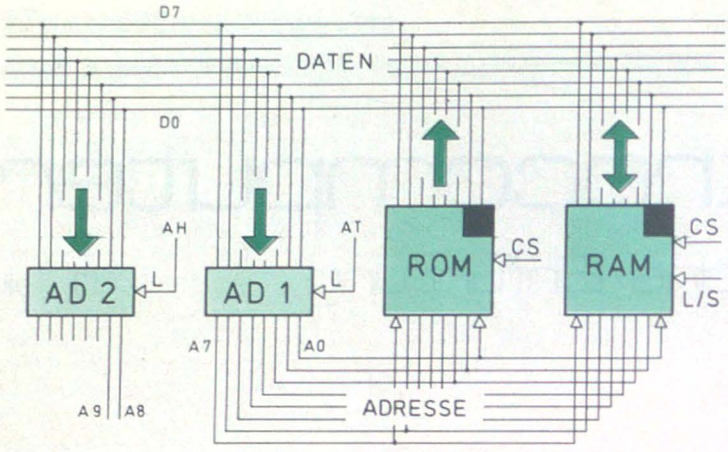
Die CPU hat nur einen Datenbus. Auf ihm werden Adresse und Daten nacheinander transferiert. Viele der heutigen Prozessoren sind dagegen mit einem getrennten Bus für Adresse und Daten ausgestattet (ähnlich Modell 1). Durch die parallele Datenverarbeitung sind höhere Geschwindigkeiten zu erzielen.

Die Adressiereingänge beider Speicher liegen parallel am Ausgang des Registers 1 (A7...A0).

Mit dem Adreßsignal A8 (Register 2) werden die Speicher ausgewählt, d. h. A8 soll über die Steuereingänge CS den gewünschten Speicher aktivieren.

A9 ist zur Steuerung der Ein- und Ausgaberegister vorgesehen. Die Signale A10...A15 werden bei dem Modell nicht benutzt.

Die Aufteilung der Adressen ist willkürlich. Andere Möglichkeiten sind denkbar. Die gewählte Lösung verschenkt sehr viele Adreßkombinationen, führt aber zu einer schaltungstechnisch einfachen Lösung.



Selektierung

...die Aufgabe der Steuerung exakter definieren. Dazu eignet sich sehr gut der Befehl LADEN, obwohl die Befehle des Modells noch nicht bekannt sind.
LA AC n bedeutet:
Lade den Inhalt der Zelle n in den Akkumulator.
n hat 16 Binärstellen und kann zwischen 000000[00][00000000] und 111111[11][11111111] liegen. Die Stellen A9...A0 sind eingerahmt.

A9	A8	A7 ... A0	
0	0	00000000	0
0	0	00000001	1
		⋮	⋮
0	0	11111111	255

A9	A8	A7 ... A0	
0	1	00000000	256
0	1	00000001	257
		⋮	⋮
0	1	11111111	511

Festlegung:

AH		AT
A9	A8	A7 ---------- A0
0	0	Speicherplatz ROM
0	1	Speicherplatz RAM

Beispiele:

A9 A8
LA AC 000000[00]00000011

Lade den Inhalt der Speicherzelle 00000011 (3) aus dem ROM in den Akkumulator.

LA AC 000000[01]00000011

Lade den Inhalt der Speicherzelle 00000011 (3) aus dem RAM in den Akkumulator.

Die CPU gibt nacheinander Adreßbyte AT und AH auf den Datenbus. Byte AT gelangt in das Register AD1 und damit an die Adreßeingänge beider Speicher. Byte AH wird in das Register AD2 geladen. Eine kleine Logikschaltung soll nun dafür sorgen, daß in Abhängigkeit von A8 und A9 (AD2) der richtige CS-Eingang auf 1 gelegt wird.

ROM
Der Programmzähler adressiert die Speicher in der Reihenfolge 0, 1, 2...Deshalb ist der untere Zahlenbereich für das ROM reserviert.
0...255: A9 = 0 A8 = 0

RAM
Die Adressen 256...511 gelten für das

RAM.
256...511: A9 = 0 A8 = 1

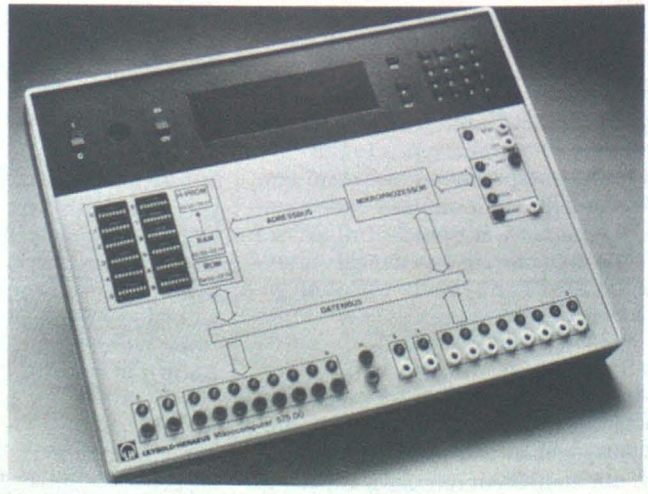
A9 ist in beiden Fällen Null und könnte fortfallen, wenn sich die Steuerung auf die beiden Speicher beschränken würde.

(Fortsetzung folgt)

Lern-System

Vor allem für die Anwendung in Schulen ist der Lerncomputer 575-00 von Leybold-Heraeus gedacht (*Bild*). Er arbeitet mit dem Mikroprozessor 8080 und einem 1-KByte-Monitorprogramm als ROM. Im Grundzustand stehen 256 Byte an RAM-Bereich zur Verfügung. Die zehn E/A-Leitungen sind über 4-mm-Buchsen zugänglich und über Schmitt-Trigger gebuffert. Anzeige und Tastatur sind für das hexadezimale Arbeiten ausgelegt. Ein Kassetten-Interface sowie ein 220-V-Netzteil sind bereits eingebaut, so daß der Mikrocomputer ohne jede Peripherie schon voll einsatzfähig ist.

Das Konzept erlaubt zu Beginn der Ausbildung die direkte Eingabe von Daten über den Datenbus, d.h. ohne das zunächst schwer verständliche Monitorprogramm. Erst später erstellt der Lernende selbst ein kleines Monitorprogramm, das schließlich von dem komfortableren Monitor-ROM abgelöst wird.



Der Mikrocomputer 575-00 ist in erster Linie für Ausbildungszwecke gedacht

Der Profi-„PET" ist da

Es ist wieder einmal soweit: Commodore wird zur Hannover-Messe '79 einen Drucker und eine Diskettenstation für den PET 2001 vorstellen.

Vor zwei Jahren wurde der Mikrocomputer PET 2001, mit vielen Vorschußlorbeeren bedacht, der Öffentlichkeit angekündigt; seit Mitte des vorigen Jahres läuft der Verkauf über rund 50 Fachhandelsgeschäfte. Bis Februar 1979 konnten etwa 4000 Stück der 8-K-Version verkauft werden. Mit der Aufnahme der Produktion im neuen Werk in Santa Clara/USA können die benötigten Stückzahlen rationeller als bisher gefertigt werden; dies hatte nun im Februar eine Preissenkung von etwa 22% zur Folge. Auch die 4-K-Version wird nun ausgeliefert.

CBM 3001 – die neue Computergeneration, mit der Commodore in den professionellen Markt vorstoßen will

Woran es bisher beim PET fehlte, ist jedem, der sich mit dem Gedanken einer Anschaffung trug, hinreichend bekannt: Es gab (und gibt) keine passende Hard- und Software von Commodore. Dies soll nun endgültig anders werden, versprach Harald Speyer, General-Manager der deutschen Niederlassung.

Nachdem man ein Jahr lang beim Drucker aufs falsche Pferd gesetzt hatte, werden heuer gleich drei Modelle vorgestellt: Ein sehr preisgünstiger elektrostatischer Drucker und zwei Matrixdrucker für Einzel- und Endlospapiertransport. Ebenso soll ein Diskettenlaufwerk mit einer Kapazität von 170 KByte auf den Markt kommen. Für die 8-K-Version soll eine Standardtastatur angeboten werden (endlich...).

Neu und speziell für den professionellen Markt zugeschnitten ist der Computer CBM 3001, den es zukünftig als 16- oder 32-KByte-Version geben wird. Der CBM 3001 ist soft- und hardwarekompatibel zum PET 2001; für den Betrieb des Diskettendoppellaufwerks sind allerdings andere ROMs erforderlich. Als „intelligentes Laufwerk" benötigt die Floppy-Disk keinen Speicherplatz im Benutzer–RAM des 3001. Auf den zwei Standard-Disks können insgesamt 360 KByte gespeichert werden. Zusammen mit einem Fortran-Compiler will man hier in neue Märkte vorstoßen. Standard-Software soll (und muß) dann natürlich ebenso angeboten werden.

Bleibt nur zu hoffen, daß aus den anläßlich einer Pressekonferenz am 22. 2. vorgestellten Mustergeräten (ein Drukker CBM 3022, ein Computer CBM 3001 und ein Floppy-Disk-Doppellaufwerk CBM 3040) in absehbarer Zeit lieferbare Einheiten werden.

Nachdem mittlerweile der Name PET von einer anderen Firma gesetzlich geschützt wurde, heißt der „PET" übrigens ab sofort „Personal Electric Transactor". –yr

Europa-Heimcomputer

Ab April wird von der schwedischen Firma Luxor, die übrigens am 2. Februar für den symbolischen Preis von einer Krone in Staatsbesitz überging, erstmals ein Heimcomputer lieferbar sein, der auch die Umlaute ä, ö und ü enthält – selbstverständlich in Groß- und Kleinschreibung. Das Gerät arbeitet, wie der TRS-80, mit der CPU Z 80, und heißt „ABC 80". Derzeit existieren noch keine festen Vorstellungen über Vertriebswege, vielmehr ist zunächst nur an Direktvertrieb gedacht.

Der ABC-80 wird in der Grundausstattung (ca. 3200 DM) mit 16 KByte ROM, 16 KByte RAM, einem speziellen, gesteuerten Kassettenrecorder und einem Schwarzweiß-Monitor geliefert (*Bild*). Er läßt sich in BASIC programmieren, mit den Befehlen PEEK und POKE können auch Programme in der Z-80-Maschinensprache eingegeben werden. Der Zeichensatz orientiert sich sehr an „Bildschirmtext", und daher läßt sich dieser Heimcomputer auch für den Zugriff auf die Zentralcomputer der Postverwaltungen einsetzen.

Die Entwicklung des Systems geschah durch die schwedische Firma Databoard, die – allerdings zu eher professionellen Preisen – bereits sehr viel Peripherie für den ABC-80 anbietet. Ein Drucker und eine Mini-Floppy-Einheit zu Preisen, die auch für den Hobbyisten interessant sind, sollen im Herbst auf den Markt kommen.

Der erste Heimcomputer mit ä, ö und ü – nur das Zeichen „ß" fehlt noch, und gegenüber der deutschen Tastatur sind Y und Z vertauscht

ASCII-Eingabe –

so ganz nebenbei

Mikrocomputer sind leider die meiste Zeit ihres Daseins damit beschäftigt, darauf zu warten, bis irgendein verarbeitungswürdiges Ereignis eintrifft, z. B. bis der Benutzer ein paar Buchstaben auf seiner Terminal-Tastatur eintippt. Daß man diese Eingabe von ASCII-Zeichen nicht zur Hauptbeschäftigung eines Mikroprozessors machen muß, zeigt das folgende Beispiel.

Die in diesem Beitrag besprochenen Programm-Routinen entstanden aus dem Wunsch, seriell eintreffende ASCII-Zeichen als einfaches Siebensegment-Alphabet auf dem sechsstelligen Display des Mikrocomputers KIM-1 darzustellen. Die Buchstaben sehen dabei zwar etwas seltsam aus, sind aber dennoch gut lesbar. Eine Anzeigeroutine für dieses Siebensegment-Alphabet wurde bereits im HOBBYCOMPUTER-Sonderheft des Franzis-Verlages abgedruckt.

Kettet man in einem Einfachst-Programm die KIM-ROM-Empfangsroutine „GETCH" und das erwähnte Siebensegment-Programm nebst einer einfachen Routine zum Verschieben der Zeichen auf dem Display aneinander, so ist der „Erfolg" deprimierend: Das Siebensegment-Display blitzt nur nach dem Empfang jeden ASCII-Zeichens kurz auf und bleibt dann wieder dunkel, da ja der Prozessor völlig damit beschäftigt ist, auf das Start-Bit des nächsten Zeichens zu warten. Bei hohen Übertragungsgeschwindigkeiten, wie etwa 600 Baud (bit/s), ist auf dem Display zwar schon etwas zu erkennen, aber erstens laufen die Buchstaben dann so schnell durch, daß man nicht mehr mitlesen kann, und zweitens treten auch noch Übertragungsfehler auf, weil die Display-Routine so lange dauert, daß bis zu ihrem Ende der Anfang des nächsten Zeichens bereits verpaßt wurde. So funktioniert es also nicht!

Die Lösung: Der Timer-Interrupt

Zum Glück verfügen viele Mikrocomputersysteme, auch der KIM-1, über einen taktgesteuerten Timer, der den Prozessor veranlassen kann, nach Ablauf einer bestimmten, programmierbaren Zeit eine Interrupt-Routine abzuarbeiten und danach in das Hauptprogramm zurückzukehren, als wäre nichts geschehen. Beim KIM-1 ist dazu die Leitung PB7 (sie ist der Interrupt-Ausgang des Timers im IC 6530-003) mit der NMI-Leitung zu verbinden.

Der Timer gliedert sich intern in zwei Teile: einen Vorteiler, der den 1-MHz-Systemtakt durch 1, 8, 64 oder 1024 teilt, und den eigentlichen Timer, der aus einem achtstufigen Binärzähler besteht. Dieser Binärzähler läßt sich per Programm auf einen beliebigen Wert zwischen Null und 255 setzen und zählt von diesem Wert aus abwärts. Ist er bei Null angelangt, so wird – wenn dieses programmiert wurde – eine negative Flanke an PB7 erzeugt, die den Interrupt auslöst.

Bild 1 zeigt das Flußdiagramm für eine Interrupt-Routine, deren Aufgabe es ist, ein ASCII-Zeichen unabhängig von einem beliebigen parallellaufenden Hauptprogramm zu empfangen und in einen Buffer abzuspeichern. Dieser Buffer wird indirekt adressiert, nämlich durch die Zellen 00EB und 00EC, wobei 00EC den höherwertigen Adreßteil enthält und 00EB den niederwertigen. Ist ein Zeichen vollständig empfangen und ist sein Wert nicht 00, so wird der niederwertige Adreßteil um 1 erniedrigt und das Zeichen an die sich dadurch ergebende neue Buffer-Adresse abgespeichert.

Diese Methode ermöglicht es, bis zu 255 Zeichen vor einer weiteren Verarbeitung durch das Hauptprogramm zwischenzuspeichern; denn schließlich werden die Zeichen u. U. langsamer ausgewertet, als sie empfangen werden, wie dies auch in unserem Anwendungsbeispiel der Fall ist.

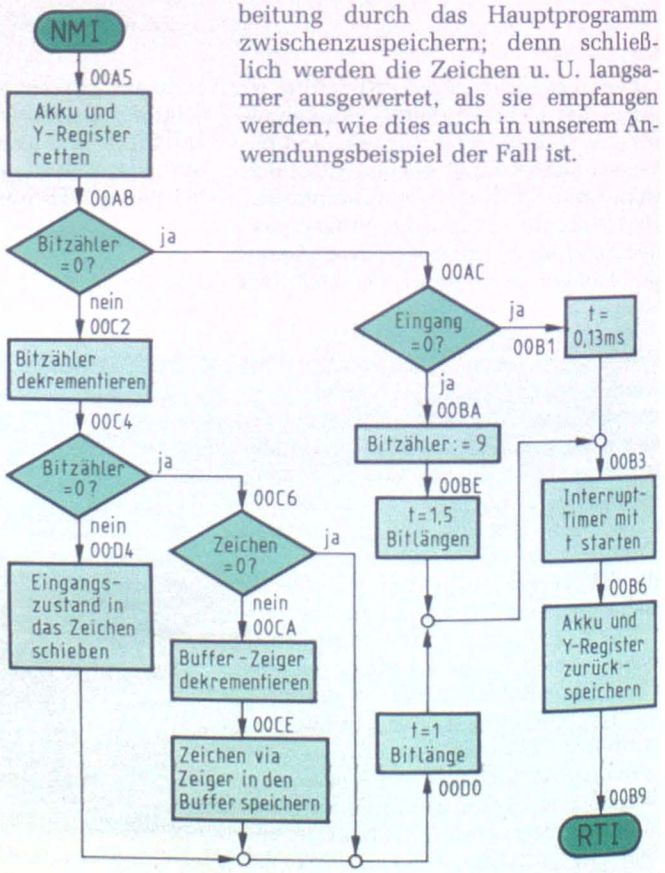


Bild 1. Flußdiagramm der Interrupt-Routine zum Empfang von seriellen ASCII-Zeichen und deren Abspeichern in einem indirekt adressierten Buffer

```
00A5 48       PHA
00A6 98       TYA
00A7 48       PHA
00A8 A5 E8    LDA E8
00AA D0 16    BNE 00C2
00AC 2C 40 17 BIT 1740
00AF 10 09    BPL 00BA
00B1 A9 02    LDA #02
00B3 8D 0E 17 STA 170E
00B6 68       PLA
00B7 A8       TAY
00B8 68       PLA
00B9 40       RTI
00BA A9 09    LDA #09
00BC 85 E8    STA E8
00BE A9 4D    LDA #4D
00C0 D0 F1    BNE 00B3
00C2 C6 E8    DEC E8
00C4 D0 0E    BNE 00D4
00C6 A5 EA    LDA EA
00C8 F0 06    BEQ 00D0
00CA C6 EB    DEC EB
00CC A0 00    LDY #00
00CE 91 EB    STA (EB),Y
00D0 A9 34    LDA #34
00D2 D0 DF    BNE 00B3
00D4 18       CLC
00D5 2C 40 17 BIT 1740
00D8 10 01    BPL 00DB
00DA 38       SEC
00DB 66 EA    ROR EA
00DD 18       CLC
00DE 90 F0    BCC 00D0
```

Bild 2. 6502-Programm als Interrupt-Routine entsprechend Bild 1

```
0000 A9 00 8D FB 17 A9 A5 8D FA 17 A2 08 A9 03 95 E6
0010 CA 10 FB 8E 0E 17 A9 7F 8D 41 17 A2 09 A0 06 84
0020 FC 98 18 65 ED 85 E6 A0 00 B1 E6 A8 C0 30 90 04
0030 C0 5B 90 02 A0 2F B9 4A 00 A0 00 8C 40 17 8E 42
0040 17 8D 40 17 A0 7F 88 D0 FD E8 E8 A4 FC 88 D0 CF
0050 20 3D 1F D0 16 C6 E9 D0 BD A9 30 85 E9 A5 ED 38
0060 E5 EB C9 FF F0 B0 C6 ED 18 90 AB 20 6A 1F C9 15
0070 10 A4 4C 00 02 00 00 00 00 00 3F 06 5B 4F 66 6D
0080 7D 07 7F 6F 00 00 00 41 00 53 00 77 7C 58 5E 79
0090 71 3D 74 05 1E 78 38 37 54 5C 73 67 50 6D 31 3E
00A0 1C 7E 76 6E 5B


0200 A2 00 8E F3 17 8E 04 17 A2 EB 8E F2 17 09 80 A2
0210 29 DD 00 02 F0 03 E8 D0 F8 E8 BD 00 02 10 03 4C
0220 00 00 20 A0 1E D0 F2 F0 F6
```

Bild 3. Dieses Programm ergibt zusammen mit der Routine aus Bild 2 einen praktischen Terminal-Tester. Startadresse ist 0000; vergessen Sie nicht, die NMI-Leitung des KIM-1 mit dem Timer-Interrupt-Anschluß PB 7 zu verbinden

Das fertige Programm

Bild 2 zeigt den disassemblierten 6502-Maschinencode für die Interrupt-Routine. Es muß noch erwähnt werden, daß sie den Interrupt-Timer selbst wieder startet; zu Beginn des Hauptprogramms muß jedoch „künstlich" ein NMI-Interrupt erzeugt werden, um die Interrupt-Routine das erste Mal anzuspringen. Das Programm in *Bild 3* tut dies; und es tut noch einiges mehr: Da, wie schon erwähnt, bei hohen Übertragungsgeschwindigkeiten die auf dem Display von rechts nach links durchlaufenden Zeichen nicht mehr mitgelesen werden könnten, wird der Zeichenbuffer recht langsam ausgelesen, nämlich mit einer Geschwindigkeit von nur etwa 3 Zeichen pro Sekunde (die genaue Zeit läßt sich in der Zelle 005A in weiten Grenzen wählen). Der „Vorlauf" der Interrupt-Eingabe-Routine kann bis zu 255 Zeichen betragen. Der Zeichenbuffer wird dabei im Adressenbereich 0300...03FF aufgebaut. Da große Teile des Hauptprogramms dem schon erwähnten Programm zur Siebensegment-Anzeige von Buchstaben sehr ähneln, wurde es lediglich in hexadezimaler Form abgedruckt. (Ausgewertet werden alle ASCII-Zeichen mit Hex-Codes zwischen 30 und 5A.)

Neben seinem ursprünglichen Verwendungszweck, nämlich für ASCII-Amateurfunkfernschreiben, läßt sich das Programm auch recht gut für den Test von Terminals und UART-Schaltungen einsetzen. Dabei lassen sich praktisch alle üblichen Übertragungsgeschwindigkeiten einstellen, wie aus *Tabelle 1* hervorgeht.

Jetzt fehlt uns für diesen Verwendungszweck also nur noch eine einfache Routine, die vorher programmierte Texte und Zeichenfolgen seriell ausgeben kann.

Das Sendeprogramm

Um auch fest programmierte Zeichenfolgen per Tastendruck über den seriellen ASCII-Anschluß des KIM senden zu können, ist im Adressenbereich 0200...0228 ein kleines Programm untergebracht, das mit Hilfe des KIM-ROM-Unterprogramms „OUTCH" Standardtexte ausgeben kann, die zwischen 0229 und 02FF stehen. Diese Standardtexte haben das Format

Tabelle 1: Wichtige Adressen im Programm

Zero-Page-Adressen

E6 Display-Zeiger L
E7 Display-Zeiger H
E8 Bit-Zähler
E9 Display-Verzögerung
EA Empfangenes Zeichen
EB NMI-Zeiger L
EC NMI-Zeiger H
ED Displ.-Zeiger-Index
FC Y-Zwischenspeicher
(Werden vom Programm gesetzt)

Geschwindigkeits-Einstellung

Baud▶	110	300	600
00BF	D4	4D	26
00D1	8E	34	1A
0201	02	00	00
0209	7D	EB	73

Tabelle 2: Hex-Code der ASCII-Zeichen

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENG	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	VS
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	Ø	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	_
6		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	'	}	~	DEL

Die waagrechten Ziffern stellen das niederwertige, die senkrechten das höherwertige Halbbyte dar; das Zeichen „e" ist z. B. hex 65. Die wichtigsten Steuerzeichen sind: CR = Wagenrücklauf (Carriage Return), LF = neue Zeile (Line Feed), BS = Cursor rückwärts (Back Space), SP = Leerraum (Space).

ID Text ID Text ID Text...FF.
ID ist dabei eine Hexzahl zwischen 80 und FE. Sie errechnet sich aus der Summe von 80 und dem Hex-Code derjenigen KIM-Taste, der der jeweilige Text zugeordnet werden soll. Das Byte FF sagt dem Computer, daß die Texttabelle hier zu Ende ist. Der Text selbst ist mit einer ASCII-Tabelle zu programmieren (*Tabelle 2*).

Das gewählte Format hat den Vorzug, daß zum Aufrufen des gewünschten Textes nicht die absolute Adresse dieses Textes bekannt sein muß. Das Auffinden geschieht vielmehr über eine Marke ID (Label). Eventuelle Änderungen der Texttabelle werden damit besonders einfach. *Herwig Feichtinger*

TRS-80
Viel Computer für wenig Geld

Heute, zwei Jahre nach ihrer Ankündigung, sind es immer noch zwei Geräte, die den Markt der betriebsfertigen Heimcomputer beherrschen: der PET 2001 von Commodore und der TRS-80 von Radio Shack (Tandy). Während der PET Vorteile im Hinblick auf Laborautomatisierung und automatische Meßwertverarbeitung aufweist (IEC-Bus), wird man den TRS-80 vorziehen, wenn man den Blick auf kleinere EDV-Projekte richtet.

Mit dem sehr leistungsfähigen Level-II-BASIC (Befehlssatz siehe *Tabelle*), 4 KByte RAM, 12-Zoll-Monitor, Kassettenrecorder und Netzteil kostet der TRS-80 rund 2100 DM. Die Tastatur entspricht etwa der einer Schreibmaschine, d. h. es ist keine unerträgliche Zumutung, wenn man z. B. über längere Zeit Daten eingeben muß. Im selben Gehäuse wie die Tastatur ist auch der komplette Computer mit der Logik für die Monitor-Ansteuerung untergebracht. Über einen Leiterplattenstecker an der Rückseite des Gerätes sind sämtliche wichtigen Signale der Z80-CPU herausgeführt. Hier wird eine sogenannte Erweiterungs-Box angeschlossen, mit deren Hilfe man die RAM-Kapazität auf insgesamt 48 KByte (im Grundgerät 16 KByte möglich) erweitern kann. Außerdem ist diese Box Voraussetzung für den Anschluß eines zweiten Kassettenrecorders, eines Mini-Floppy-Disk-Laufwerks (bis zu vier möglich) oder eines Druckers. Wer sich in der Schaltungstechnik von Mikroprozessoren etwas auskennt, der kann an den Erweiterungsstecker aber auch einen Peripherie-Baustein (PIO, PIA oder ähnliches) anschließen, um Ein/Ausgabe-Kanäle zur Verfügung zu haben. Bedauerlich ist, daß man in der Grundausführung ohne zusätzlichen Aufwand keine externen Einheiten oder Schaltungen steuern kann.

Der gesamte Computer ist so ausgelegt, daß die CPU alle Aufgaben übernimmt, die man ihr zumuten kann. Diesem Konzept, das natürlich aus Preisgründen gewählt wurde, ist z. B. eine genormte serielle Schnittstelle (RS232, erst mit Erweiterungs-Box möglich) zum Opfer gefallen, die den Datenaustausch mit Geräten anderer Hersteller ermöglichen würde (der Grund ist die parallele Abfrage der Tastatur). Zusätzlich muß der Mikroprozessor Teile der Video-Interface-Logik übernehmen, und er läuft „nur" mit einer Taktfrequenz von 2 MHz, obwohl 4 MHz beim Z80 möglich wären. Das alles trägt dazu bei, daß das TRS-80-BASIC nicht zu den schnellsten gehört. Trotzdem ergab der Vergleich mit einem 6800-BASIC, daß der TRS-80 Sinusberechnungen etwa um den Faktor 9 schneller ausführt.

Der TRS-80 ist inzwischen zum meistverkauften „Personal Computer" geworden

Auffallend im Befehlssatz sind einerseits sehr leistungsfähige Editierbefehle, die das Arbeiten mit dem Computer wesentlich erleichtern, andererseits Grafikbefehle, mit denen man Einzelpunkte setzen, löschen und abfragen kann. Der Bildschirm faßt im Normalbetrieb 16 Zeilen zu 64 Zeichen, im Grafik-Modus können 128 Punkte horizontal und 48 Punkte vertikal aufgelöst werden; die Position wird in einem X-Y-Koordinatensystem angegeben. Mathematisch orientierte Anwender werden das einem Satz von Sonderzeichen mit geringerer Auflösung vorziehen.

Für den späteren Anschluß des Floppy-Disk-Laufwerkes, zu dem es ein Disketten-Betriebssystem gibt, sind bereits die entsprechenden Befehle vorhanden. In beschränktem Maße lassen sich auch mit dem Kassettenrecorder Dateien aufbauen.

Ein Wort noch zu Problemen, die beim Betrieb auftauchen: Mitunter gibt es Schwierigkeiten mit Fertigkassetten, die man inzwischen in großer Zahl erwerben kann. Das liegt im allgemeinen daran, daß die Kopfstellung des Aufnahmegerätes mit der des eigenen Recorders nicht übereinstimmt. Es emp-

fiehlt sich, die Justierschraube für die Kopfverstellung zugänglich zu machen und dann auf beste Höhenwiedergabe abzugleichen. Mit dieser Methode war beim Verfasser bisher jedes Programm lesbar.

Ein weiteres Problem läßt sich leider nicht so einfach beheben: Bei längerem Betrieb (1...2 Std.) machen plötzlich verschiedene Speicher nicht mehr mit. Die Folge ist, daß man das Gerät abschalten muß und erst wieder weitermachen kann, wenn die Chips abgekühlt sind. Ein typisches Wärmeproblem, das sich nur durch gute Entlüftung umgehen läßt. Ähnliche Effekte sind uns auch vom PET bekannt. Man sollte sich von diesen Unzulänglichkeiten aber nicht zu sehr zur Kritik verleiten lassen: Es ist und bleibt eine erstaunliche Tatsache, daß man diese Leistung zu einem derart geringen Preis bekommt. *Rudolf Hofer*

µC-Nachrichten

Ein 8-KByte-BASIC für den Mikrocomputer **SYM-1** ist von Electronic 2000 (München) zu haben, und zwar in Form zweier ROM-ICs, die in zwei der SYM-Leersockel passen und den Adressenbereich C000...DFFF belegen. Als Arbeitsspeicher wird der Adressenbereich ab 0200 verwendet. Das Komplettsystem SYM-1 mit dem Videointerface KTM-2 und den BASIC-ROMs kostet dann etwa 2300 DM. Der BASIC-Befehlsvorrat entspricht weitgehend dem PET 2001, die Ausgabegenauigkeit beträgt 8 Stellen.

□

Auch **Siemens** plant, in den Heimcomputer-Bereich vorzudringen und entwickelt derzeit ein System mit dem Mikroprozessor 6502. Dies ist besonders deshalb interessant, weil Siemens damit von der sonst vertretenen Intel-Produktgruppe abweicht und einen „fremden" Prozessor einsetzt.

□

Das Heinz-Piest-Institut für Handwerkstechnik an der Universität Hannover führt nun offiziell den neuen Elektronik-**Fachlehrgang** IV C „Mikrocomputer" durch. Am 15. Januar hatten bereits 15 Schulungsstätten ihre Anerkennung für diesen Fachlehrgang erhalten, deren Adressen beim Heinz-Piest-Institut, Wilhelm-Busch-Straße 18, 3000 Hannover 1, zu erfragen sind.

Tabelle: **Befehle und Eigenschaften von Level-II-BASIC**

Variablen-Typen

$	String (0...255 Zeichen)
%	Integer (–32768 to 32767)
!	Einfache Genauigkeit (6 Stellen angezeigt)
#	Doppelte Genauigkeit (16 Stellen angezeigt)

Arithmetische Funktionen

ABS	PEEK
ATN	POINT
CDBL	POS
COBL	RANDOMIZE
CINT	RND
COS	SGN
CSNG	SIN
ERL	SPC
ERR	SQR
EXP	TAB
FIX	TAN
MEM	USR
INP	VARPTR
INT	
LOG	

Programm-Befehle

CMD	GOSUB	POKE
DEFDBL	GOTO	REM
DEFINT	IF	RESET
DEFSNG	LET	RESTORE
DEFSTR	NEXT	RESUME
DIM	ON ERROR GOTO	RETURN
END	ON...GOTO	SET
ERROR	ON...GOSUB	STOP
FOR	OUT	

Ein/Ausgabe-Befehle

INPUT	DATA	CLOAD	PRINT #
PRINT	SET	CSAVE	INPUT #
READ	RESET	INKEY $	PRINT USING

Kommandos

CLEAR	DELETE	LIST	NEW	TROFF
CONT	EDIT	LLIST	RUN	TRON

String-Funktionen

ASC	MID$
CHR$	RIGHT$
LEFT$	STR$
LEN	VAL

Disk-Befehle

E/A-Befehle		E/A-Funktionen	
CLOSE	LSET	CVD	LOF
DATA	OPEN	CVI	MKD$
FIELD	PRINT	CVS	MKI$
GET	PUT	EOF	MKS$
KILL	RESET	LOC	

Spezielle TRS-80-Befehle

SET	POINT	CLOAD	RANDOMIZE
RESET	INKEY	CSAVE	CMD

Editier-Befehle

A – Gehe zum Zeilenanfang
nC – n Zeichen ändern
nD – n Zeichen löschen
E – Ende des Editiervorgangs
H„Text" – Rest der Zeile löschen und „Text" einfügen
I„Text" – „Text" einfügen
L – Rest der Zeile drucken und zum Zeilenanfang gehen
Q – Alle Änderungen ignorieren
und einige andere

Programm zur Spulenberechnung mit dem TI-59

Bei der Dimensionierung von Spulen greift der Hf-Techniker meist eher zum Lötkolben als zum Taschenrechner – das Umrechnen von Induktivitätswerten in Windungszahlen ist denkbar kompliziert. Programmierbare Taschenrechner machen daraus allerdings ein Kinderspiel; das Programm selbst ist gegen Einsenden einer leeren Magnetkarte oder auch in Form von Drucker-Auflistungen von der FUNKSCHAU-Redaktion (Postfach 370120, 8000 München 37) erhältlich. Das Problem der Spulenberechnung reduziert

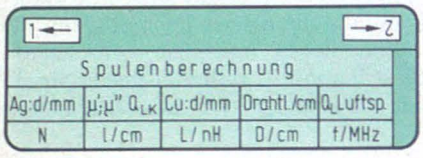


Bild 1. Gegen Einsendung eines frankierten Rückumschlages erhalten Sie das TI-59-Programm zur Spulenberechnung

sich dadurch auf das Einschieben einer Magnetkarte in den Rechnerschlitz (*Bild 1*). Die gesuchte Größe N, l, L oder D wird einfach Null gesetzt. Nach Eingabe von d, µ' und µ'' sowie nach Anzeige von µ'' muß R/S gedrückt werden.

Das Programm eignet sich für einlagige Zylinderspulen mit oder ohne Kern. Folgende Berechnungen sind möglich:

- Errechnen der Induktivität einer Spule bekannter Windungszahl und bekannter Abmessungen.
- Für eine gewünschte Induktivität kann – bei bekannten Spulenkörper-Abmessungen – die notwendige Windungszahl berechnet werden.
- Berechnung der Spulendaten bei gegebener Induktivität für ein bestimmtes Platinen-Raster.

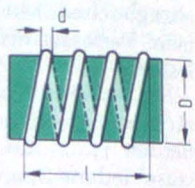
Induktivität, Windungszahl, Spulenlänge und Spulendurchmesser berechnen sich nach folgenden Gleichungen:

$$L = \frac{\pi^2 \cdot D^2 \cdot N^2}{l + 0{,}45\,D - 0{,}003\,\frac{D^2}{l}}\ [\mathrm{nH}]$$

$$N = \sqrt{\frac{L}{D^2 \cdot \pi^2}\left(l + 0{,}45\,D - \frac{0{,}003 \cdot D}{l}\right)}$$

$$l_{Sp} = -0{,}5\left(0{,}45\,D - \frac{\pi^2 \cdot D^2 \cdot N^2}{L}\right) + \sqrt{\left(0{,}5\,(0{,}45 \cdot D - \frac{\pi^2 \cdot D^2 \cdot N^2}{L})\right)^2 + 0{,}003 \cdot D^2}\ [\mathrm{cm}]$$

Bild 2. ▶ Mechanische Abmessungen der Spule



$$D = \frac{0{,}45 \cdot 0{,}5 \cdot L}{\pi^2 \cdot N^2 \cdot 0{,}003\,\frac{L}{l}} + \sqrt{\left(\frac{0{,}45 \cdot 0{,}5 \cdot L}{\pi^2 \cdot N^2 \cdot 0{,}003\frac{L}{l}}\right)^2 + \frac{L \cdot l}{\pi^2 \cdot N^2 - 0{,}003\frac{L}{l}}}\ [\mathrm{cm}]$$

Dabei ist
N = Windungszahl,
l_{Sp} = Spulenlänge in cm,
D = Spulendurchmesser in cm,
L = Induktivität in nH,
l_D = benötigte Drahtlänge cm,
d = Drahtdurchmesser in mm,
δ = Eindringtiefe in µm,
α = spez. Widerstand,
A' = äquivalente Stromfläche.

Darüber hinaus kann das Programm über nachstehende Formeln folgende Werte berechnen:

$$l_D = N \cdot \pi \cdot \sqrt{\left(\frac{1}{N-1}\right)^2 \cdot D^2 + 1\ \mathrm{cm}}$$

(1 für 2mal 0,5 cm Anschlußdrähte)

$$Q_{LL} = \frac{\omega L}{R_L}\text{, wobei } R_L = \frac{l_D}{\varkappa \cdot A'};$$

$$A' = \delta \cdot d \cdot \pi \text{ u. } \delta_{Cu} =$$

$$\frac{67}{\sqrt{f\ \mathrm{MHz}}} \text{ bzw. } \delta_{Ag} = \frac{69{,}5}{\sqrt{f\ \mathrm{MHz}}}$$

$$Q_{LK} = \frac{1}{\frac{R_L}{\mu' \cdot \omega \cdot L} + \frac{\mu''}{\mu'}}$$

Hierbei ist
Q_{LL} = Güte der Luftspule,
Q_{LK} = Güte der Spule mit Kern,
µ' = μ_{rel} des Kerns,
µ'' = Verlustbeiwert des Kerns.

Bild 2 zeigt, wie die Formelbuchstaben für die mechanischen Abmessungen der Spule zu verstehen sind.

Ing.(grad.) Fred H. Heller

16-bit-Bausatzsystem

Unter der Bezeichnung SDK-86 ist von Intel (Distributor: Alfred Neye-Enatechnik GmbH) ein „System-Design-Kit" als Bausatz lieferbar. Alles, was für das 8086-System benötigt wird, ist in dem Bausatz enthalten. Als Speicher werden 2 KByte des Intel-Bauelementes 2142 (stat. 4-K-RAM) verwendet, mit den dazugehörigen Sockeln, um die Kapazität zu verdoppeln. Es ist genügend Platz vorhanden für 8 KByte Programmspeicher. Zwei 4-K-ROM-residente Monitore für Tastatur oder Teletype/Datensichtgerät werden ebenfalls mitgeliefert.

Um den Umgang mit dem SDK-86 zu vereinfachen, sind verschiedene Möglichkeiten zur Ein- und Ausgabe vorgesehen. Ferner ist ein Kabel zusätzlich verfügbar, das das SDK-86 mit jedem Intellec-System verbindet.

Als Datenunterlagen werden das Assembly- und Users-Manual für SDK-86 sowie das „8086 Users/Manual" mitgeliefert.

Die auf der Platine befindliche LED-Anzeige umfaßt acht Siebensegment-Displays. 24 Tasten gestatten die Programmeingabe im Hexadezimal-Code. Für Kundenschaltungen steht auf der Platine eine frei verdrahtbare Wire-Wrap-Fläche zur Verfügung. Als Versorgungsspannungen werden 5V (3,5A) und 12V (0,3A) benötigt – letztere allerdings nur bei Anschluß eines ASCII-Fernschreibers. Dessen Baud-Rate kann 110...4800 Bd betragen.

Hans Werner

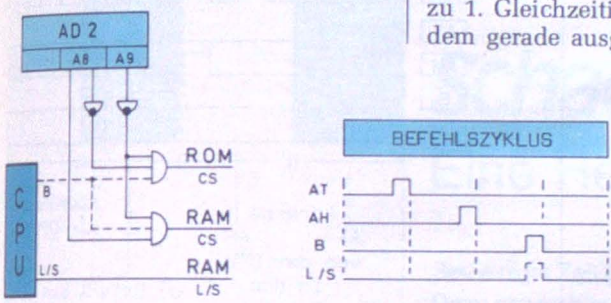
Mikrocomputer

Eine kleine Einführung

14. Teil

Steuerung

Die erforderliche Logikschaltung zur Steuerung der Speicher ist sehr einfach. Es genügen 2 UND-Gatter und 2 Inverter.



ROM

Der CS-Eingang des ROM ist 1, wenn die Adreßsignale A8, A9 gleich 0 sind und der Impuls B gleich 1 ist. Impuls B verhindert unerwünschte Zwischenergebnisse, denn erst nach Zeitpunkt AH sind beide Adreßteile in den Registern AD1 und AD2 gespeichert.

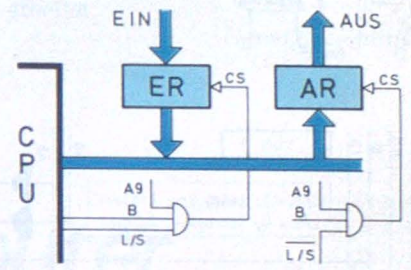
RAM

Ist A8 gleich 1 und A9 gleich 0, wird mit Impuls B der CS-Eingang des RAM zu 1. Gleichzeitig liegt, abhängig von dem gerade ausgeführten Befehl, entweder L/S = 0 (Schreiben) oder L/S = 1 (Lesen) am Speicher.

Ein- und Ausgabe

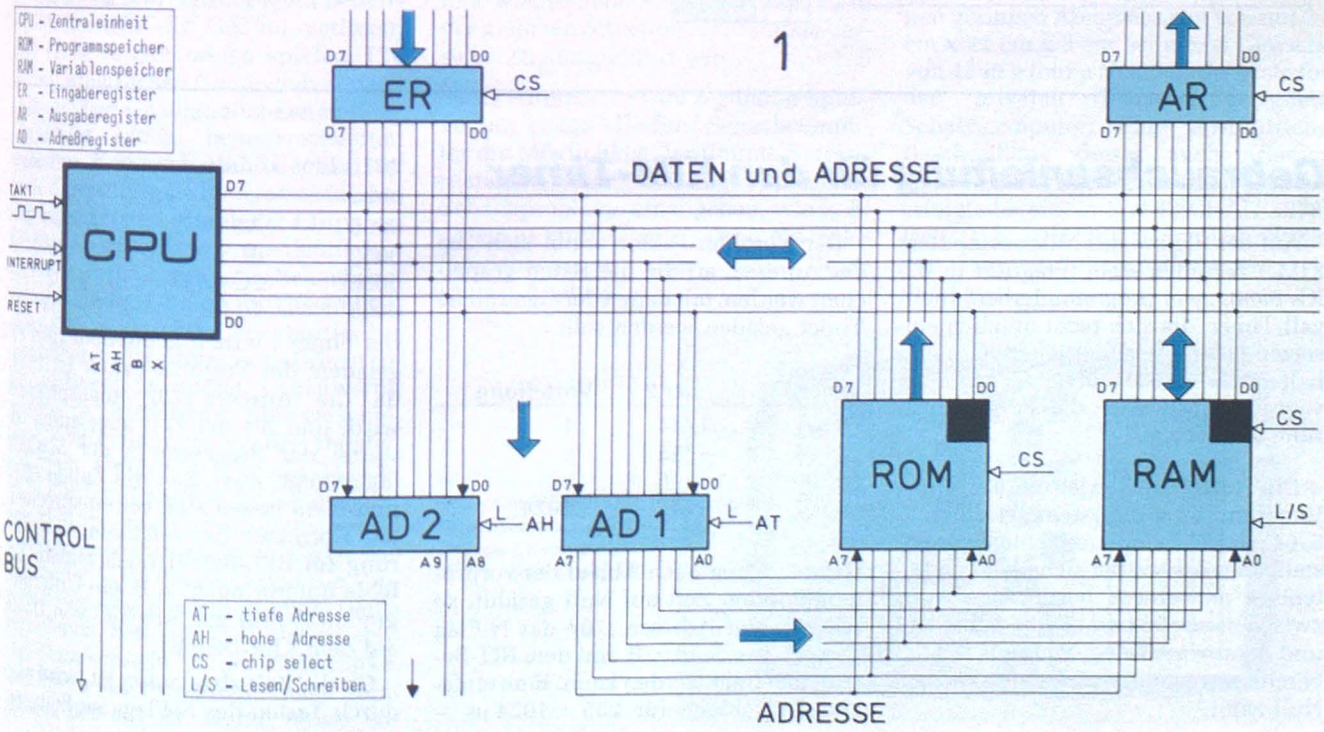
Daten müssen ein- und ausgegeben werden. Wie bei Modell 1, geschieht das mit zwei Registern. Unser Modellcomputer hat keine speziellen Ein- und Ausgabebefehle.

Eingaberegister ER und Ausgaberegister AR werden mit Adreßstelle A9 angesprochen. Die Auswahl erfolgt mit L/S, die Synchronisierung wieder mit Impuls B.



Die CPU will Daten lesen:
L/S wird 1 und mit Adresse 512 (A9 = 1) auch CS/ER.
Die Daten gelangen vom Eingaberegister ER in die CPU.

Die CPU gibt Daten aus:
L/S wird 0. Die Daten werden aus der CPU in das Ausgaberegister AR geladen, vorausgesetzt A9 ist 1.



Zusammenfassung

ROM: Adresse 0...255 (A9 = 0, A8 = 0)
RAM: Adresse 256...511 (A9 = 0, A8 = 1)
Ein-Ausgabe: Adresse 512...XXX (A9 = 1, A8 = X)

Alle Adressen zwischen 512 und 1023 sprechen die Ein-Ausgaberegister an. In diesem Bereich ist A9 auf jeden Fall 1. Es ist aber sinnvoll, nur eine Möglichkeit zu verwenden; wir einigen uns auf 512.

Was folgt noch?

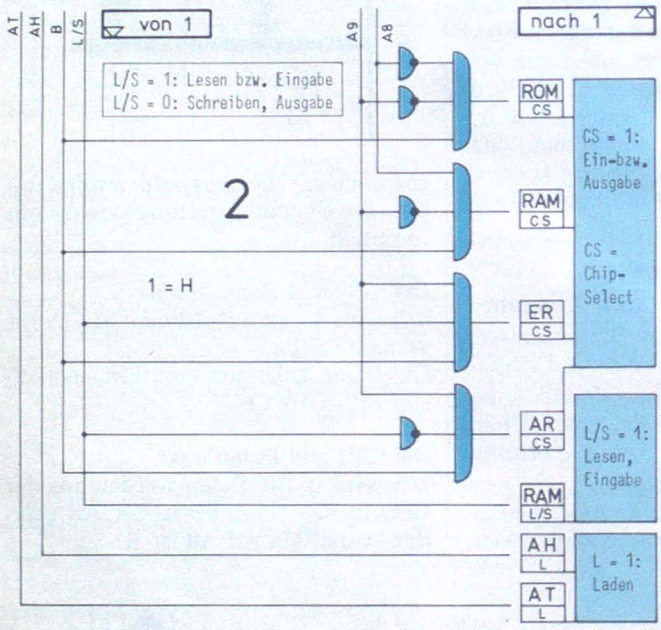
- Verschiedene Speichertypen
- Befehlsabläufe des Modells
- Ein üblicher Befehlssatz
- Das Modell ist schon fast ein Profi
- Unterprogramme
- Interruptroutinen

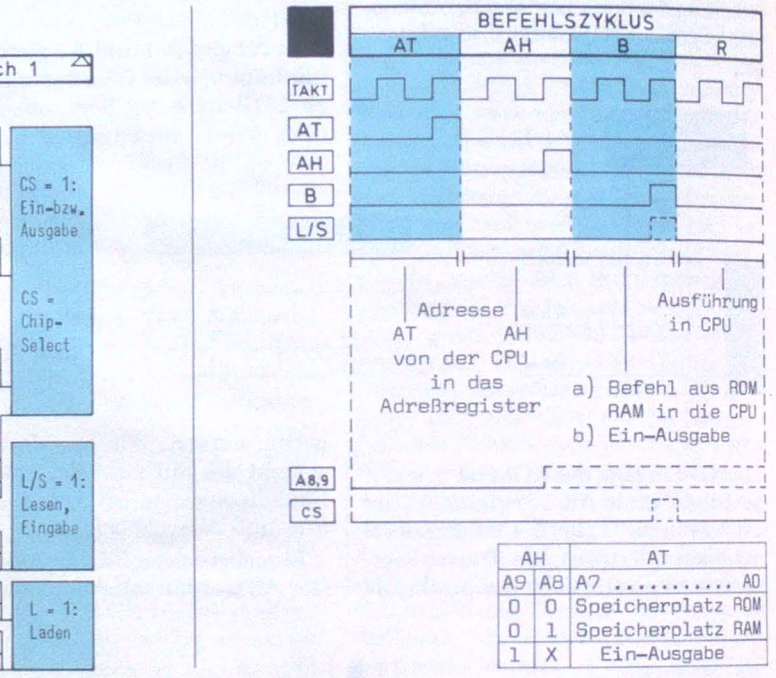
Auf der vorhergehenden Seite steht die vollständige Schaltung. Daten und Adresse werden auf einem einzigen, bidirektionalen Bus transportiert.

Die Adresse umfaßt 2 x 8 = 16 bit. Sie wird in 2 Teilen zu 8 bit in den Registern AD1 und AD2 gespeichert.

Die hohe Adresse AH (Bits A8 bis A15) wird bei der gewählten Außenschaltung lediglich zur Selektierung der Speicher und der Ein-Ausgabe-Register benutzt.

(Fortsetzung folgt)





AH		AT
A9	A8	A7 ..... A0
0	0	Speicherplatz ROM
0	1	Speicherplatz RAM
1	X	Ein-Ausgabe

Gebrauchsanleitung für den KIM-Timer

Auf der Platine des Mikrocomputers KIM-1 befinden sich, integriert in die ICs 6530, zwei programmierbare Intervall-Timer, die sich recht nützlich einsetzen lassen. Bedauerlicherweise enthalten die KIM-Handbücher nur recht vage Angaben über die Programmierung der Timer.

Die Timer bestehen jeweils aus einem Vorteiler, der den Systemtakt durch 1, 8, 64 oder 1024 teilt, und einem voreinstellbaren Zähler, der sich über den Datenbus mit einem bestimmten Wert zwischen hex 00 und hex FF laden läßt und dann von diesem Wert aus mit der bereits vorgeteilten Frequenz bis auf Null zählt.

Der Vorteiler-Faktor ergibt sich aus der Adresse, an die die Daten gespeichert werden, mit denen der eigentliche Timer geladen werden soll:

Timer 1	*Timer 2*	*Vorteilung*
1704	1744	1
1705	1745	8
1706	1746	64
1707	1747	1024

Hat der Timer nach Ablauf der vorprogrammierten Zeit auf Null gezählt, so wird an der Adresse 1707 das N-Flag gesetzt, das dann z. B. mit dem BIT-Befehl abgefragt werden kann. Eine einfache Zeitschleife für 255 · 1024 µs = 261,12 ms könnte dann z. B. so aussehen:

```
0050    A9 FF     LDA #FF
0052    8D 07 17  STA 1707
0055    2C 07 17  BIT 1707
0058    10 FB     BPL 0055
```

Der Timer 1 wird hier mit dem Wert FF geladen; der Vorteiler-Faktor ist 1024, da die Adresse 1707 angesprochen wird; und bis die Zeit abgelaufen ist, wartet der Prozessor in der Schleife 0055/0058. Zwischen die Zeilen 0052 und 0055 lassen sich bei entsprechender Korrektur der relativen Adressierung im BPL-Befehl noch andere Befehle unterbringen, z. B. ein Unterprogramm-Aufruf zur Anzeige von Daten auf dem KIM-Display.

Ob die Zeit abgelaufen ist, wird also durch Testen des N-Flags im Statusre-

gister an der Adresse 1707 erkannt. Will man dagegen den aktuellen Zählerstand auslesen, so kann das z. B. mit dem Befehl LDA 1706 geschehen, also stets an der Adresse 1706.

Einer der beiden KIM-Timer, nämlich Timer 1, ist in der Lage, bei Ablauf der vorprogrammierten Zeit einen Interrupt auszulösen. Addiert man zu den oben angegebenen Adressen jeweils 0008, so werden die Vorteiler-Faktoren wie oben eingestellt, nach dem Ablauf der gewünschten Zeit wird jedoch am Anschluß PB 7 (der als Eingang programmiert sein muß) eine negative Flanke ausgelöst.

Verbindet man nun PB 7 z. B. mit der NMI-Leitung, so läßt sich der Timer 1 so programmieren, daß nach Ablauf einer gewissen Zeit ein NMI-Interrupt ausgelöst wird. Die Befehlsfolge zum Starten des Timers kann hier z. B. lauten:

```
0050    A9 FF     LDA # FF
0052    8D 0F 17  STA 170F
```

In diesem Beispiel tritt nach 261,12 ms ein nicht maskierbarer Interrupt auf. Verwendet man die gleiche Befehlsfolge in der Interrupt-Routine selbst, so entsteht ein periodischer Interrupt. Dann ist jedoch das Initialisieren durch diese Befehlsfolge auch zu Beginn des Hauptprogramms erforderlich, um den Interrupt das erste Mal auszulösen. Die Methode, den Timer als periodische Interrupt-Quelle zu verwenden, ist sehr universell anwendbar und ersetzt externe Interruptquellen wie etwa RC-Impulsgeneratoren. Sie bietet auch den Vorteil, daß die Interrupt-Frequenz praktisch quarzstabil ist, da der Timer mit dem quarzgesteuerten Systemtakt arbeitet.

Herwig Feichtinger

Björn Schwarz

Schachcomputer

Eine Herausforderung besonderer Art

Seit einiger Zeit gibt es auf dem deutschen Markt eine Vielzahl von Computerspielen. Dazu gehören auch die Schachcomputer, die inzwischen zu einem großen Verkaufsschlager geworden sind. Der nachfolgende Bericht bietet eine Übersicht über die derzeit erhältlichen Geräte.

Schachcomputer sind sowohl für Anfänger als auch Fortgeschrittene als Lehrmeister hervorragend geeignet, da sie praktisch alle Schachregeln beherrschen, niemals die Geduld verlieren und recht variantenreich spielen. Die derzeit erhältlichen Geräte haben nach anfänglichen Unzulänglichkeiten ihrer Vorgänger einen bemerkenswerten technischen Stand erreicht, der in erster Linie durch die hohe Integrationsdichte der verwendeten Halbleiter-Chips ermöglicht wurde. Durch die damit verbundene Verbesserung des Schachprogramms sind sie einem durchschnittlichen Spieler durchaus ebenbürtig, bei den höheren Spielstärken teilweise sogar überlegen geworden. Gegenwärtig werden in Deutschland fünf verschiedene Schachcomputer von drei Herstellerfirmen angeboten (*Tabelle 1*). Obwohl alle fünf Geräte mit leistungsfähigen Mikroprozessoren ausgerüstet sind und auch ausnahmslos Spezialzüge wie „Rochade", „En Passant" und die Umwandlung des Bauern in eine Dame vornehmen, bestehen mitunter erhebliche Unterschiede in der Spielstärke, was durch das verwendete Schachprogramm bedingt ist. Durch einen Zufallsgenerator, mit dem alle Schachcomputer bis auf den „Chess Challenger 3" bestückt sind, ergeben sich immer wieder neue Spielvarianten, da in der gleichen Situation nicht immer derselbe Zug ausgeführt wird.

Als Alternative zum regulären Spielverlauf bieten alle fünf Schachcomputer die Möglichkeit, bestimmte Spielsituationen zur Lösung spezieller Schachprobleme einzugeben, wie z. B. „Matt in zwei Zügen" sowie ein Telefon-Schachspiel oder eine Briefschach-Analyse.

Chess Challenger 10

Der Vorgänger dieses Schachcomputers, „Chess Challenger 3", war mit seinen geringen Abmessungen von nur 34 cm x 21 cm x 3 cm bei einem Gewicht von 1240 g (ohne Netzteil) der erste für den privaten Gebrauch geeignete Schachcomputer. Eine ausführliche Beschreibung dieses auch „Senior Chess Challenger" genannten Gerätes erfolgte bereits in FUNKSCHAU 1978, Heft 21, S. 1045, so daß an dieser Stelle von weiteren Angaben abgesehen werden kann.

Bild 1. Seit Herbst 1978 ist der Chess Challenger 10 als verbesserte Version des Chess Challenger 3 erhältlich

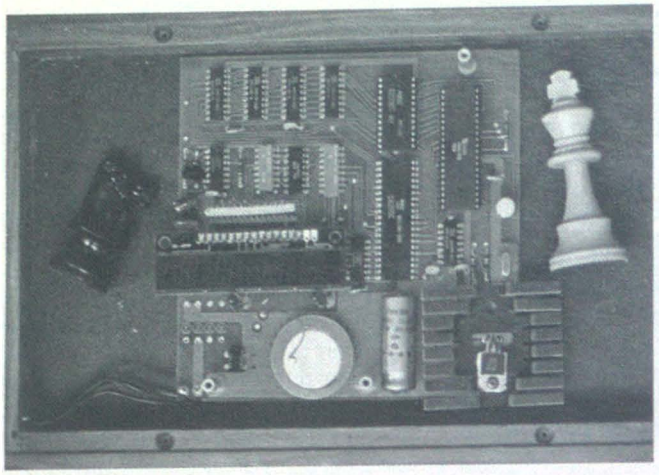


Bild 3. Der Chess Champion MK 1 gehört zu den preiswerteren Schachcomputern

◄ Bild 2. Innenleben des Chess Challenger 10

Verfügte der „Chess Challenger 3" nur über drei einstellbare Spielstärken, so besitzt die seit dem Herbst 1978 erhältliche verbesserte Version „Chess Challenger 10" (*Bild 1*) insgesamt 10 Programmstufen mit 6 Spielstärken und 4 Analysearten. Die durchschnittlichen „Denkpausen" des Gerätes betragen 5 s (Stufe 1), 15 s (Stufe 2), 35 s (Stufe 3), 80 s (Stufe 4), 140 s (Stufe 5) und 3 min (Stufe 10).

Im Verlauf des Spiels kann die Spielstärke jederzeit geändert und eine Überprüfung der Stellung sämtlicher Figuren vorgenommen werden. Durch Eliminierung oder Umstellung der Schachfiguren kann jede beliebige Mittel- oder Endspielsituation eingegeben und in den vier Analysestufen durchgespielt werden. Aufgrund der Bestükkung mit einem Mikroprozessor des Typs Z 80A (*Bild 2*) besitzt der „Chess Challenger 10" im Mittelspiel einen großen Einfallsreichtum, wobei er pro Zug bis zu 3 024 000 Stellungen analysiert.

Chess Champion MK I

Dieser von vielen Kaufhäusern angebotene Schachcomputer „Chess Champion MK I" (*Bild 3*) wird leider nur mittleren Ansprüchen gerecht, weist jedoch von allen Schachcomputern das beste Preis/Leistungsverhältnis auf. Das Gerät verfügt über 6 Programmstufen, die sich in 4 Spielstufen mit unterschiedlichem Schwierigkeitsgrad und 2 Analysestufen einteilen lassen. Während des Spiels läßt sich die Spielstärke ändern, und auch die Überprüfung der Stellung aller Schachfiguren ist möglich. Die mittleren Antwortzeiten des „Chess Champion MK I" sind 1 s, 15...20 s, 15 s...15 min und 20 s...15 min (je nach Stufe). In Stufe 6 (Analyse-Briefschach) dauert es jedoch unter Umständen 2 Tage, bis der Gegenzug auf der vierstelligen LED-Anzeige erscheint.

Wesentliche Nachteile des „Chess Champion MK I" bestehen darin, daß er nur mit den schwarzen Figuren spielt und nicht nur erlaubte, sondern auch regelwidrige Züge akzeptiert. Obwohl er damit nicht die Ansprüche guter Schachspieler erfüllt, ist er dennoch für Anfänger und Fortgeschrittene als Lernhilfe geeignet.

Tabelle 1: Derzeit erhältliche Schachcomputer

Merkmal	*BORIS*	*Chess Challenger 10*	*Chess Champion MK I*	*Compu Chess II*
spielt schwarz oder weiß	ja	ja	nein (nur schwarz)	ja
spielt gegen sich selbst	ja	nein	nein	nein
Seitenwechsel während der Partie möglich	ja	nein	nein	nein
Figurenanzeige in Symboldarstellung	ja	nein	nein	nein
Anzeige von Alternativzügen	ja	nein	nein	nein
alphanumerische Anzeige (Kommentar)	ja	nein	nein	nein
Anzahl der Programmstufen	>100	10	6	6
Anzahl der für beide Seiten „vorausgedachten" Züge	4	3	3	3
einfache Fehlerkorrektur	ja	nein	nein	nein
automatische Bauernbeförderung	ja	ja	nein (manuell)	nein (manuell)
automatische Rochade	ja	ja	nein (manuell)	nein (manuell)
automatisches „En Passant"	ja	ja	nein (manuell)	nein (manuell)
„Springerspiel" möglich	nein	nein	nein	ja
„Amazonenkönigin" möglich	nein	nein	nein	ja
mitgeliefertes Schachspiel	ja	ja	nein	nein
Preis (Stand 15.2.1979) ca.	898 DM	698 DM	248 DM	ca. 600 DM

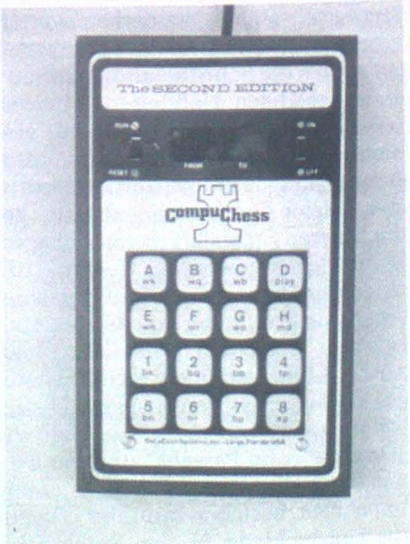


Bild 4. Der CompuChess II ist eine Weiterentwicklung des Chess Champion MK 1

CompuChess II

Der seit kurzer Zeit erhältliche „CompuChess II" (*Bild 4*) stellt die verbesserte Version des „Chess Champion MK I" dar. Das Gerät spielt sowohl mit den schwarzen als auch mit den weißen Figuren und weist etwas kürzere Antwortzeiten auf. Für den Gegenzug benötigt der „CompuChess II" nunmehr 1 s (Stufe 1), ca. 7 s (Stufe 2) und 10 s bis 9 min (Stufe 3 und Stufe 4). Als Novität beherrscht das Gerät auch die Spielarten „Springerspiel" und Amazonenkönigin". Beim „Springerspiel" werden Dame, Läufer und Türme durch Springer ersetzt, so daß jede Partei über 8 Bauern, einen König und 7 Springer verfügt. Bei der Spielart „Amazonenkönigin" können die beiden Damen wahlweise auch wie Springer ziehen.

Trotz dieser zusätzlichen Spielvarianten, über die sonst keiner der getesteten Schachcomputer verfügt, wird der „CompuChess II" jedoch den Ansprüchen überdurchschnittlicher Schachspieler nicht gerecht, weil er wie sein Vorgänger „Chess Champion MK I" ebenfalls regelwidrige Züge akzeptiert. Da für knappe 100 DM mehr der wesentlich besser spielende „Chess Challenger 10" erhältlich ist, kann der Kauf des „ChompuChess II" nicht empfohlen werden.

BORIS

Seit Dezember 1978 gibt es auch in Deutschland den nach Expertenmeinung gegenwärtig leistungsfähigsten Schachcomputer „BORIS" (*Bild 5*). Einem internationalen Bewertungssystem zufolge, nach dem z. B. Bobby Fischer 2800 Punkte hätte, ein Großmeister 2500 bis 2700, Internationale Meister bis 2300 und starke Clubspieler bis zu 1800 Punkte aufweisen, bringt es „BORIS" auf respektable 1600 Punkte.

„BORIS" spielt wahlweise die weißen oder schwarzen Figuren oder sogar gegen sich selbst, wodurch er dem menschlichen Gegenspieler aus einer bedrohlichen Situation heraushelfen kann. Mittels einer programmierbaren Uhrenschaltung für die Einstellung beliebiger Antwortzeiten zwischen 1 s und 100 Std. ergeben sich mehrere unterschiedliche Spielstärken. Der Schachcomputer kann bis zu vier Züge für beide Seiten im voraus „denken" und vermittelt seinem Gegenspieler Kenntnis über die von ihm verfolgte Strategie durch Anzeige auch der von ihm erwogenen, jedoch nicht durchgeführten Spielzüge.

Bild 5. Nicht gerade billig, aber stark im Spiel: Der Schachcomputer BORIS

Eine Besonderheit stellen 80 verschiedene Kommentare dar, die gelegentlich auf der achtstelligen alphanumerischen Anzeige in Laufschrift angezeigt werden, wie z. B. „IS THIS A TRAP?", „I EXPECTED THAT", „MATERIALIST" u. a.. Weitere Eigenschaften des in einem geschmackvollen handpolierten Edelholzgehäuse zusammen mit den Schachfiguren und einem zusammenfaltbaren Schachbrett untergebrachten, leicht bedienbaren Schachcomputers sind: Rücknahme des letzten Spielzuges, Änderung der Spielstärke während des Spiels, Symbolanzeige der Stellung sämtlicher Figuren, programmierbare Figurenaufstellung für die Analyse von Schachproblemen und Lösung von Mattsituationen in zwei Zügen.

Einblick in die von „BORIS" verfolgte Strategie vermittelt die gegen einen Durchschnittsspieler durchgeführte Schachpartie in *Tabelle 2*, wobei die mittlere Antwortzeit 30 Sekunden betrug.

Zur Befriedigung der Neugier, welcher Mikroprozessor und welche periphere Beschaltung, insbesondere zur Speicherung der diversen Kommentare, bei „BORIS" zum Einsatz gelangte, wurde das Gerät geöffnet. Doch welche Überraschung! Statt des erwarteten Mikroprozessors Z 80, 8080, 8085 oder F8 kam die kundenspezifische Schaltung AMI 7839 SAA zum Vorschein.

Vergleich der Schachcomputer

Zur Feststellung der tatsächlichen Spielstärke aller derzeit erhältlichen Schachcomputer wurde ein kleines

Tabelle 2. Typischer Spielverlauf mit BORIS

Antwortzeit 30 s			
Zug-Nr.	*Kommentar*	*BORIS (weiß)*	*Mensch (schwarz)*
	BORIS AWAITS YOUR MOVE		
1		E2–E4	G8–F6
2		D2–D3	D7–D5
3		F2–F3	B8–C6
4	UNLEASHING THE ATTACK	C2–C3	A7–A5
5		B1–A3	B7–B6
6		D1–A4	C8–D7
7		A4–B5	E7–E6
8		E4xD5	E6xD5
9		G2–G4	F8xA3
10		B2xA3	C6–D4
11		B5–B2	F6xG4
12		C3xD4	D8–H4 +
13		E1–D2	G4–F2
14		F1–G2	F2xH1
15	WHY DO YOU DO THAT?	G2xH1	H4xH2 +
16		G1–E2	H2xH1
17		B2–C3	E8–C8 ≙ 0–0–0
18		F3–F4	H1–F3
19	WHERE DID I GO WRONG?	D2–E1	D7–H3
20		A1–B1	D8–E8
21		B1–B2	F3–F1 +
22	THIS NEEDS CARE CONGRATULATIONS	E1–D2	F1–E2 ∓

Zeichenerklärung:

–	ziehe Figur von ... nach ...	+	Schach
x	schlage Figur	∓	Schachmatt
		0–0–0	lange Rochade

Computer-Schachturnier ausgetragen. Die ersten Spiele bestritt „BORIS" gegen den „Chess Champion MK I". Da „BORIS" sämtliche Zweikämpfe gewann, wurde sein Gegner für die Endrunde nicht berücksichtigt.

Bei den anschließend durchgeführten Spielen zwischen dem „Chess Challenger 10" und dem „ChompuChess II" zeigte sich der „Chess Challenger 10" seinem Gegner ebenfalls weit überlegen, so daß sich der „CompuChess II" für die Teilnahme an der Endrunde nicht qualifizieren konnte.

Waren „Chess Champion MK I" und „CompuChess II" für „Chess Challenger 10" und „BORIS" noch keine ernstzunehmenden Gegner, so lieferten sich die beiden letztgenannten Schachcomputer teilweise spannende Duelle.

Die „BORIS"-Zeitschaltung wurde vorher wiederum zur Gewährleistung der Chancengleichheit auf die den unterschiedlichen Spielstärken des „Chess Challenger 10" zugeordneten mittleren Antwortzeiten eingestellt. Da diese jedoch vom „Chess Challenger 10" im weiteren Verlauf des Turniers häufig überschritten wurden, ließ sich ein echter Vergleich der Leistungsfähigkeit beider Schachcomputer nicht durchführen.

Entsprechend der Struktur des Schachprogramms spielt der „Chess Challenger 10" in der Eröffnungsphase recht einfallslos, benötigt dafür allerdings auch nur kurze „Denkpausen", so daß er die zur Verfügung stehende Antwortzeit nicht ausnutzt. In der Mittelphase der Schachpartie spielt der Chess Challenger 10 „ideenreich" und greift bevorzugt mit vielen Figuren meist vom Rand des Schachbretts aus an, wobei er beim Ausschalten der gegnerischen Figuren den Schutz der eigenen nicht vernachlässigt. Der Chess Challenger 10 benötigt in dieser Spielphase viel Zeit und überschreitet häufig die angegebene mittlere Antwortzeit.

Bei den niedrigen Spielstärken reagiert der Chess Challenger 10 im Endspiel teilweise enttäuschend schwach, arbeitet jedoch bei den Programmstufen 4, 5 und 10 zielstrebiger auf das Matt hin, wofür er jedoch auch wieder mehr Zeit benötigt.

Auf die Spielstärke von „BORIS" wirkt sich die genaue Einhaltung der Antwortzeiten negativ aus, da er für die Entwicklung einer guten Strategie oftmals nicht genügend Zeit hat. Während „BORIS" nur einen bis zu diesem Zeitpunkt ermittelten Gegenzug ausführen kann, erscheint die Antwort des „Chess Challenger 10" erst dann auf der Anzeige, wenn er sämtliche durch das Schachprogramm vorgeschriebenen Stellungsanalysen abgeschlossen hat. Die zur Verfügung stehende mittlere Antwortzeit wird dadurch oftmals überschritten.

Dieser Unterschied zwischen den Schachprogrammen von „BORIS" und „Chess Challenger 10" wirkte sich beim Schachturnier nachteilig für „BORIS" aus. Bei den kurzen Antwortzeiten bis ca. 35 s erwies sich „BORIS" dennoch dem „Chess Challenger 10" als ebenbürtig und war bei sehr kurzen Antwortzeiten (Blitzschach) seinem Gegner sogar überlegen.

Das Turnier wurde beendet durch Schachpartien mit der höchsten Spielstärke, wobei der „Chess Challenger 10" auf Programmstufe 10 (mittlere Antwortzeit 3 min) und „BORIS" jeweils auf die vom „Chess Challenger 10" tatsächlich benötigte Antwortzeit (2,5 bis ca. 8 min) eingestellt wurde.

Auf die Spielstärke von „BORIS" wirkte sich diese Maßnahme derart vorteilhaft aus, daß er nunmehr die meisten Spiele zu seinen Gunsten entscheiden konnte.

Als Fazit des Schachturniers kann festgestellt werden, daß ein Leistungsvergleich der beiden Schachcomputer nicht möglich ist, weil die von ihnen verwendeten Schachprogramme voneinander abweichende Strukturen aufweisen, wodurch die Geräte nach unterschiedlichen Strategien vorgehen.

Da „BORIS" und „Chess Challenger 10" den anderen Schachcomputern weit überlegen sind, können sie Freunden des Schachspiels für die Bereitung vieler schöner Stunden sehr empfohlen werden. Die Entwicklung von Schachcomputern mit höherer Leistungsfähigkeit ist sicherlich möglich, sie wird jedoch noch einige Zeit in Anspruch nehmen.

Ein Software-Tondecoder

Tondecoder sind Schaltungen, die das Vorhandensein eines Tones bestimmter Frequenz erkennen. Für diesen Zweck sind zwar preisgünstige PLL-ICs auf dem Markt, es ist aber dennoch reizvoll, eine solche Tonerkennung rein auf Software-Basis durchzuführen.

Das Programm (*Bild 1*) wurde für den Mikrocomputer KIM-1 mit der CPU 6502 geschrieben, auf den sich auch die Adresse des I/O-Ports (PB 3) bezieht, an dem die zu prüfende Frequenz anliegt. Die Startadresse des Beispiel-Programms ist 0000.

Die Software-Tondecodierung besitzt einige deutliche Vorteile gegenüber herkömmlichen Hardware-Lösungen. So ist etwa die „Resonanzfrequenz" ebenso stabil wie der für die Mikrocomputer-Taktsteuerung verwendete Quarz; Frequenz und Bandbreite lassen sich in weiten Grenzen beliebig wählen, und der Zeitraum, der für das Decodieren eines Tones erforderlich ist, ist stets nur gerade so groß, wie es die programmierte Bandbreite erfordert.

Bild 2 macht die Arbeitsweise des Programms deutlich. Es wartet zunächst auf eine negative Flanke der Eingangsfrequenz; tritt eine solche innerhalb einer Periodendauer der Sollfrequenz nicht auf, so wird das Programm trotzdem fortgesetzt, um nicht „hängenzubleiben".

Dann wird nach einem Viertel der Soll-Periodendauer der Zustand des Eingangs abgefragt. Bei der richtigen Frequenz muß er noch Null sein, andernfalls erfolgt sofort ein Rücksprung.

Anschließend wird nun um eine halbe Soll-Periodendauer verzögert. Bei der richtigen Frequenz muß der Zustand des Eingangs dann 1 sein. Nach einer weiteren halben Periodendauer wird überprüft, ob der Eingang mittlerweile Null ist, und das Spiel wiederholt sich so oft, bis eine vorher programmierte Zahl von Perioden der Sollfrequenz durchlaufen ist. Entspricht der Zustand des Eingangs nun immer noch der „Vorhersage" des Mikrocomputers, so handelt es sich um die richtige Frequenz.

```
TONDECODER-UNTERPROGRAMM
0040 A0 2D     LDY #2D
0042 A9 08     LDA #08
0044 88        DEY
0045 F0 06     BEQ 004D
0047 2C 02 17  BIT 1702
004A F0 F8     BEQ 0044
004C 88        DEY
004D F0 05     BEQ 0054
004F 2C 02 17  BIT 1702
0052 D0 F8     BNE 004C
0054 A0 05     LDY #05
0056 A2 16     LDX #16
0058 CA        DEX
0059 D0 FD     BNE 0058
005B 2C 02 17  BIT 1702
005E D0 0F     BNE 006F
0060 A2 2D     LDX #2D
0062 CA        DEX
0063 D0 FD     BNE 0062
0065 2C 02 17  BIT 1702
0068 F0 05     BEQ 006F
006A A2 2D     LDX #2D
006C 88        DEY
006D D0 E9     BNE 0058
006F 98        TYA
0070 60        RTS
BEISPIEL: EINSCHALTEN DES
KIM-DISPLAY BEI DER SOLL-
FREQUENZ
0000 20 40 00  JSR 0040
0003 D0 03     BNE 0008
0005 20 1F 1F  JSR 1F1F
0008 4C 00 00  JMP 0000
```

Bild 1. 6502-Maschinenprogramm zur Tondecodierung

Die Bandbreite (B), innerhalb der ein Ton richtig erkannt wird, läßt sich mit der Zahl der überprüften Perioden (M) frei wählen. Bezeichnet man die Sollfrequenz mit f_o, so gilt die Beziehung

$$M = \frac{f_o}{2\,B}.$$

Da M (Adresse 0055) die hexadezimalen Werte 01...00 annehmen kann – 00 wird hier als dezimal 256 interpretiert –, ist die Güte (f_o/B) zwischen 2 und 512 frei wählbar. Bei einer Sollfrequenz von 1 kHz läßt sich daher eine Bandbreite von 2...500 Hz erzielen.

Die Frequenz wird mit den beiden hexadezimalen Variablen a (0057) und b (0061, 006B, 0041) festgelegt, die sich folgendermaßen errechnen:

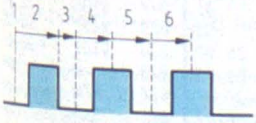


Bild 2. Prinzip der periodischen Abtastung. 1 = Programmstart, 2 = Warten bis zur nächsten negativen Flanke, 3 = Verzögerung um 0,25 Perioden, 4...6 = Verzögerung um je eine halbe Periode

$$a = \frac{1}{5}\left(\frac{10^6}{4\,f_o} - 7\right) \qquad b = \frac{1}{5}\left(\frac{10^6}{2\,f_o} - 9\right)$$

f_o kann dabei etwa zwischen 400 Hz und 30 kHz liegen (µP-Taktfrequenz: 1 MHz). Die Dimensionierung in der Programmauflistung erfolgte für eine Sollfrequenz von 2125 Hz und eine Bandbreite von rund 210 Hz. Die Decodierzeit ist von der Differenz zwischen der anliegenden und der Soll-Frequenz abhängig und beträgt maximal, d. h., wenn beide Frequenzen übereinstimmen, etwa

$$T = \frac{M + 2}{f_o}.$$

In unserem Beispiel ergibt sich ein Wert von etwa 3,3 ms. Es ist leicht einzusehen, daß die Decodierzeit der Bandbreite etwa umgekehrt proportional ist.

Bei der Rückkehr aus dem Unterprogramm sind Akku und Y-Register Null, wenn die richtige Frequenz erkannt wurde, und das Z-Flag im Status-Register ist 1. Das X-Register enthält dann den Wert b. Wurde nicht die Sollfrequenz decodiert, so ist das Z-Flag Null; der Akku und das Y-Register enthalten dann die Anzahl der Zyklen, die für ein richtiges Dekodieren noch gefehlt hatten. Damit ist auch eine Aussage darüber möglich, wie weit die Frequenz am Eingang vom Sollwert abweicht. Der Inhalt des X-Registers ist Null.

Das beschriebene Prinzip der Tondecodierung durch periodisches Abtasten wird auch als Sample-Methode bezeichnet. Zwei kleine, meist allerdings vernachlässigbare Nachteile des Prinzips sollen nicht verschwiegen werden: Der für ein richtiges Decodieren erforderliche Störabstand beträgt rund 6...10 dB, bezogen auf weißes Rauschen. Die Mikrocomputer-Tonerkennung ist damit guten PLL-Schaltungen um rund 6 dB unterlegen, was aber nur in Ausnahmefällen eine Rolle spielt. Den zweiten Nachteil hat die beschriebene Methode mit PLL-Schaltungen gemeinsam: Es gibt mehrere Frequenzen, die zu einem Ansprechen des Decoders führen können. So spricht das Sample-Prinzip auch auf 5 f_o, 9 f_o, 13 f_o usw. an, allerdings mit herabgesetzter Empfindlichkeit und Bandbreite. Im Gegensatz zur PLL werden jedoch keine „Subharmonischen" ausgewertet, Frequenzen also, deren Oberwellen zufällig die Sollfrequenz ergeben.

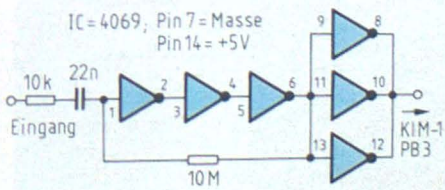


Bild 3. Einfache Vorverstärkerschaltung

Selbstverständlich wird am Eingang des Mikrocomputers eine saubere Rechteckspannung mit möglichst symmetrischem Tastverhältnis benötigt. *Bild 3* zeigt eine geeignete Schaltung zur Aufbereitung von Eingangssignalen ab einigen mV.

Herwig Feichtinger

Keine weitere Datenverarbeitungs-Produktförderung

Mit dem Jahr 1978 lief das dritte Datenverarbeitungs-Förderungsprogramm der Bundesregierung aus. Von den 3,2 Mrd. DM an Förderungsmitteln hatte Siemens allein schon immerhin 1 Mrd. erhalten, womit nicht nur 4000 neue Arbeitsplätze geschaffen werden konnten, sondern sich Siemens auch einen Marktanteil von etwa 20 % auf dem EDV-Markt verschaffte. Damit ist Siemens zumindest in der Bundesrepublik hinter dem Giganten IBM (60 % Marktanteil) auf dem zweiten Platz.

Wie Bundesforschungsminister Volker Hauff bei einer Besichtigung des Siemens-Werkes München-Perlach (Unternehmensbereich Daten- und Informationssysteme) ankündigte, will die Bundesregierung für die Produktentwicklung auf dem EDV-Gebiet nun keine Förderungsmittel mehr zur Verfügung stellen, weil der technologische Rückstand gegenüber den USA aufgeholt sei.

Sicher spielte bei der Verbesserung der deutschen Situation auch die deutsch-japanische Zusammenarbeit eine Rolle; während Siemens ein Großrechner-Programm von Fujitsu für Deutschland übernahm, zeigte umgekehrt Fujitsu großes Interesse an dem von Siemens entwickelten Laserdrukker, von dem bereits große Stückzahlen verkauft wurden.

Statt der Förderung der EDV-Produktentwicklung möchte die Bundesregierung jetzt lieber bessere „Umweltbedingungen" für die Industrie schaffen, z. B. durch Finanzierung von Datenverarbeitungs-Projekten im öffentlichen Bereich.

HF

H. Becher

Nebenempfangsstellen schnell berechnet

Das nachfolgend aufgeführte Programm soll erläutern, wie durch das gezielte Eliminieren von unnötigen Operationen der Rechenablauf von langsam arbeitenden Taschenrechnern beschleunigt werden kann.

Entstehung und Berechnung von Nebenempfangsstellen sind bekannt und sollen hier nicht behandelt werden. Programme für Taschenrechner sind bereits veröffentlicht worden, z. B. für den Rechner TI 59 in FUNKSCHAU 1978, Heft 23, Seite 1171. Bei diesem Programm werden die Störfrequenzen

$$f_{stör} = \frac{n \cdot f_{osz} \pm f_{ZF}}{m} \qquad (1)$$

für alle Kombinationen von m und n berechnet und anschließend der Vergleich

$$f_u < f_{stör} < f_o$$

durchgeführt.

Für Berechnungen, wo einerseits die Grenzen f_u und f_o eng gewählt werden können und andererseits der Oszillatormultiplikationsfaktor n groß ist (z. B. für Funkgeräte im UHF-Bereich), führt der Rechner unnötig viele Berechnungen aus, was sich auf die schnelle Verfügbarkeit der Resultate negativ auswirkt.

In dem hier vorliegenden Programm, ausgelegt für die Rechner HP 67/97, werden nur die Störfrequenzen im interessierenden Bereich f_u bis f_o berechnet und angezeigt *(Bild 1)*. Für jedes m wird deshalb zuerst das minimale n berechnet. Aus (1) folgt:

$$n_{min} = \text{int}\,\frac{m \cdot f_u - f_{ZF}}{f_{osz}} + 1 \qquad (2)$$

Die Berechnung erfolgt nun, wie das Listing in *Bild 2* zeigt, mit Formel (1),

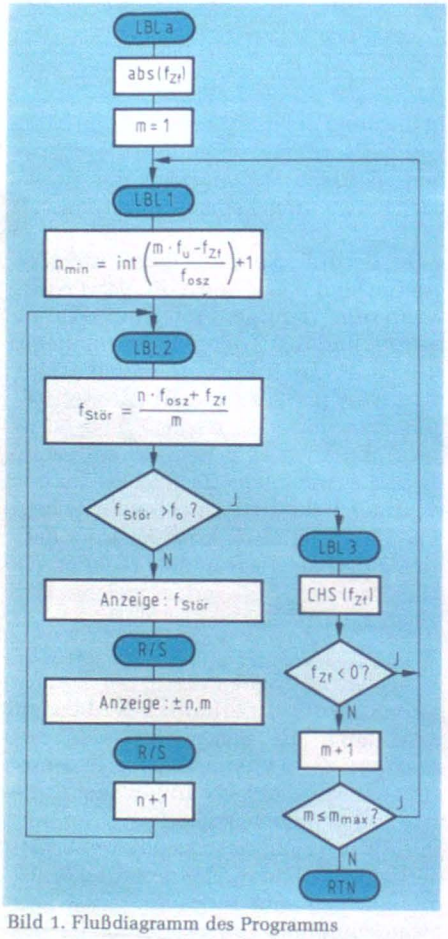


Bild 1. Flußdiagramm des Programms

und zwar mit $n = n_{min}, n_{min} + 1$, usw., solange bis f_o überschritten wird. (Eine Eingabe von n_{max} erübrigt sich deshalb ebenfalls).

Anschließend wird das Vorzeichen im Speicher f_{ZF} gewechselt, und es können mit den gleichen Formeln (2) und (1) die Störfrequenzen für $-f_{ZF}$ errechnet werden. Die Einsparung an Rechenzeit gegenüber konventionellen Programmen kann bis zu etwa 95 % betragen.

Die Rechenzeit kann durch geeignete Maßnahmen noch weiter verkürzt werden. Dazu wird aber eine größere Anzahl Speicherregister und Programmschritte benötigt. Das vorliegende Programm kann der Kürze wegen mit kleinen Anpassungen auch auf dem HP-65 verwendet werden.

Bedienung des Rechners

Schritt	Bedienung	Eingabe	Taste	Anzeige
1	Magnetkarte einlesen			0.000000
2	Oszillatorfrequenz eingeben	f_{osz}	A	f_{osz}
3	Zwischenfrequenz eingeben	f_{Zf}	B	f_{ZF}
4	Untere Grenzfrequenz eingeben	f_u	C	f_u
5	Obere Grenzfrequenz eingeben	f_o	D	f_o
6	Max. Eingangs-Harmonische	m_{max}	E	m_{max}
7	Rechnungsbeginn		fa	
8	Anzeige der Störfrequenz			$f_{stör}$
9	Abruf der Vervielfachungszahlen		R/S	$\pm n \cdot m$
10	Fortsetzung der Berechnung bei Schritt 8		R/S	
11	Ende der Berechnung			0.000000

```
001  *LBLA      21 11
002   STO1      35 01
003    R/S         51
004  *LBLB      21 12
005   STO2      35 02
006    R/S         51
007  *LBLC      21 13
008   STO3      35 03
009    R/S         51
010  *LBLD      21 14
011   STO4      35 04
012    R/S         51
013  *LBLE      21 15
014   STO5      35 05
015    R/S         51
016  *LBLa   21 16 11
017   DSP6     -63 06
018   RCL2      36 02
019    ABS      16 31
020   STO2      35 02
021      1         01
022   STO8      35 08
023  *LBL1      21 01
024   RCL8      36 08
025   RCL3      36 03
026      x        -35
027   RCL2      36 02
028      -        -45
029   RCL1      36 01
030      ÷        -24
031    INT      16 34
032      1         01
033      +        -55
034   STO7      35 07
035  *LBL2      21 02
036   RCL4      36 04
037   RCL7      36 07
038   RCL1      36 01
039      x        -35
040   RCL2      36 02
041      +        -55
042   RCL8      36 08
043      ÷        -24
044    X>Y?      16-34
045   GTO3      22 03
046    R/S         51
047   RCL7      36 07
048   RCL8      36 08
049    EEX        -23
050      6         06
051      ÷        -24
052      +        -55
053   RCL2      36 02
054   ENT↑       -21
055    ABS      16 31
056      ÷        -24
057      x        -35
058    R/S         51
059      1         01
060   ST+7   35-55 07
061   GTO2      22 02
062  *LBL3      21 03
063   RCL2      36 02
064    CHS        -22
065   STO2      35 02
066    X<0?      16-45
067   GTO1      22 01
068   RCL5      36 05
069   RCL8      36 08
070      1         01
071      +        -55
072   STO8      35 08
073    X≤Y?      16-35
074   GTO1      22 01
075      0         00
076    RTN         24
077    R/S         51
```

Bild 2. Programm für die Rechner HP-97 und HP-67

Hans Werner

Mikrocomputer

Eine kleine Einführung

15. Teil

Die CPU gibt die Adresse 00000001 00000111 heraus. Welche Speicherzelle wird angesprochen? ☐ a) Zelle 7 im ROM ☐ b) Zelle 263 im RAM ☐ c) Zelle 7 im RAM	bei Lösung a) nach J b) nach N c) nach K	A
Fein, die Antwort ist richtig. Schwer zu beantworten war die Frage allerdings nicht.	nach C	B
Die Adreßbits A9, A8 werden zur Selektierung der Speicher und der Ein- Ausgaberegister benutzt. Bei 2 Speichern können nur $2 \times 2^8 = 2 \times 256 = 512$ Speicherplätze adressiert werden, obwohl der Programmzähler $2^{16} = 65535$ (0...65535) Speicherzellen adressieren könnte. Wie sieht die Steuerung aus, wenn mehr als 512 Speicherplätze zu adressieren sind?	nach M	C
Das stimmt nicht. Die Adresse gelangt zwar in zwei Hälften auf den Datenbus, wird aber in den Adreßregistern gespeichert. Dadurch stehen zu einem bestimmten Zeitpunkt alle 16 bit gleichzeitig zur Verfügung. *Richtige Lösung:* In einem der Speicher (ROM) ist das Programm gespeichert. Es soll auch nach einem Stromausfall erhalten bleiben. Der andere Speicher (RAM) wird als Zwischenspeicher für Daten und Ergebnisse benutzt. Er verliert bei Stromausfall seinen Inhalt. Aus dem ROM können nur Daten ausgelesen werden.	nach C	D
Unser Modell hat einen gemeinsamen Bus für Daten und Adresse. Was hat das für Nachteile? ☐ a) Daten und Adresse können nicht gleichzeitig transferiert werden. ☐ b) Der gemeinsame Bus ermöglicht größere Arbeitsgeschwindigkeiten. ☐ c) Der Steueraufwand ist relativ groß.	bei Lösung a) nach O b) nach G c) nach P	E
Das ist leider nicht richtig. Suchen Sie sich eine neue Lösung aus.	nach Q	F
Die Lösung ist nicht richtig. Versuchen Sie erneut Ihr Glück.	nach E	G

Sie haben richtig erkannt, daß die 16stellige Adresse außerhalb der CPU gespeichert werden muß. Nur so stehen alle 16 bit gleichzeitig zur Verfügung.

nach I **H**

Der Steuerbus führt 4 Signale: AT AH L/S und B

Ein Signal ist von der Art des gerade ausgeführten Befehls abhängig, die anderen 3 nicht. Welches Signal ist das?

nach S **I**

Schade, die Antwort ist falsch. Versuchen Sie es noch einmal.

nach A **J**

Sie haben richtig gewählt. Mit A9 = 0 und A8 = 1 wird das RAM angesteuert. Die anderen 8 bit geben die Speicherzelle an. 00000111 = 7

nach L **K**

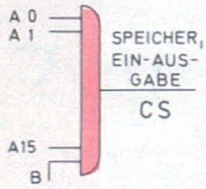
Ein Mikrocomputer hat normalerweise zwei verschiedene Speicher. Warum?

☐ a) Ein Speicher soll bei Stromausfall die Daten (Programm) behalten, der andere jederzeit veränderlich sein.

☐ b) Weil die Adresse in zwei Hälften zu 8 bit auf den Datenbus gelangt.

bei Lösung
a) nach B
b) nach D

L



Ganz einfach. Es können z. B. die 4 höchsten Adreßkombinationen zur Steuerung herausgegriffen werden.

ROM: 65532
ROM: 65533
Eingaberegister: 65534
Ausgaberegister: 65535

Wir haben das schon so bei Modell 1 gemacht.

Erinnern Sie sich?

nach E **M**

Nein, lieber Leser. Das ist nicht richtig. Zurück zur Aufgabe. Achten Sie besonders auf die Stellen 8 und 9.

nach A **N**

Sie haben recht. Zusätzlich ist noch Lösung c richtig (großer Steueraufwand).

nach Q **O**

Das stimmt. Natürlich ist auch Lösung a richtig (Daten und Adresse nicht gleichzeitig).

nach Q **P**

Warum sind die beiden Adreßregister erforderlich?

☐ a) Weil der Datenbus nur 8 bit hat.
☐ b) Weil A8 und A9 zur Steuerung benutzt werden.
☐ c) a und b treffen zu.

bei Lösung
a) nach H
b) nach F
c) nach R

Q

Die Antwort ist falsch. Sie haben sich verwirren lassen. Zurück zur Aufgabe.

nach Q **R**

Haben Sie es gewußt? Es ist das Signal L/S.

Das war die letzte Aufgabe. Aber nicht frohlocken: Es folgen noch weitere Tests!

PAUSE **S**

(Fortsetzung folgt)

Sergio Menchinelli

Jedem Finger seinen Synthesizer

Polyphoner Musik-Synthesizer mit Mikroprozessor

Mehrstimmigkeit bei geringem Preis – das ist das Ziel dieses mikrocomputergesteuerten Musikinstruments. Der Mikroprozessor ist der SCMP von National Semiconductor. Er wird über eine Klaviertastatur angesteuert und liefert Ausgangssignale für zehn digital steuerbare monophone Synthesizer.

Der polyphone Synthesizer ist der vorläufige Schlußpunkt einer Entwicklung, die in den sechziger Jahren mit dem Entwurf spannungsgesteuerter Oszillatoren [1] begann und sich dann in den Siebzigern mit Instrumenten fortsetzte, welche die elektronische Manipulation der Einhüllenden als wesentlichem Teil eines musikalischen Ereignisses ermöglichten. Neuere Veröffentlichungen über polyphone Synthesizer [2, 3, 4] bieten interessante Lösungen für die digitale Ansteuerung einer Anzahl von spannungsgesteuerten Synthesizer-Moduln. Als Extremfall hat Ferguson [5] eine Art „Synthesizer für jede Taste" vorgeschlagen, da ihm die digitale Steuerung zu kompliziert und zu teuer erschien. Das entwickelte System benutzt den SCMP-Prozessor von National Semiconductor als Steuergerät für die Kontrolle über 10 elektronische Synthesizer-Moduln. – Jedes dieser Moduln enthält die folgenden Baugruppen:

a) spannungsgesteuertes Filter
b) digital gesteuerter Oszillator
c) spannungsgesteuerter Verstärker
d) Einhüllendengenerator.

Der digital gesteuerte Oszillator ist im Prinzip ein programmierbarer Teiler. Eine alternative Form, der selbstinkrementierende Akkumulator, entwickelte sich zum Kernstück eines neuen monophonen Synthesizers, der hervorragende Abstimmeigenschaften besitzt.

Systemübersicht

Wie in *Bild 1* dargestellt ist, besteht das Instrument aus einer Klaviertastatur mit 61 Tasten (entspricht 5 Oktaven), einem Steuergerät und zehn einstimmigen Synthesizern, deren Ausgangsspannungen aufsummiert die gesamten Töne von bis zu zehn gleichzeitig gedrückten Tasten wiederzugeben vermögen.

Jeweils 8 Tasten werden über einen entprellenden 8-bit-Eingangsport (2 x DM 8544) auf den 8-bit-Datenbus geschaltet. Es werden somit nur 8 Leseoperationen benötigt, um die Tastatur vollständig abzufragen. Der DM 8544 eignet sich besonders für diese Anwendung, da er direkt an den Bus angekoppelt werden kann und über interne pull-up-Widerstände zur Tastenentprellung verfügt. Für die Eingangsschaltung werden also außer der Tastatur, bestückt mit einpoligen Umschaltern, nur noch 16 DM 8544 ohne zusätzliche aktive oder passive Bauelemente benötigt.

Die Ausgangsports des Steuergeräts bestehen aus je zwei CMOS-Registern 74 C 175, die den Eingängen der peripheren einstimmigen Synthesizern zugeordnet sind. In ihnen steht das Steuerbyte mit der in *Bild 2* dargestellten Zuordnung: Die vier niedrigstwertigen Bits C0/bis C3 repräsentieren den Tonhöhencode (Note) innerhalb einer Oktave, während die drei Bits C 4, C 5, C 6 die Oktave angeben. Schließlich ist unabhängig von C 0 bis C 6 das MSB C 7 genau dann auf 0, wenn eine Taste gedrückt ist. Diese Unabhängigkeit ist nicht selbstverständlich, denn die Tonhöheninformation wird ja noch gebraucht, um den Ton mit der richtigen Frequenz ausklingen zu lassen.

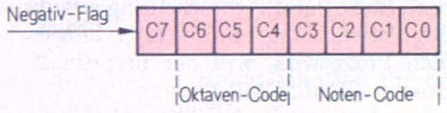


Bild 2. Steuerwort für die Synthesizer

Softwareaufbau

Im Prinzip genügt es, bei Tasteninstrumenten für jeden Finger einen Synthesizer vorzusehen: mehr als zehn Ta-

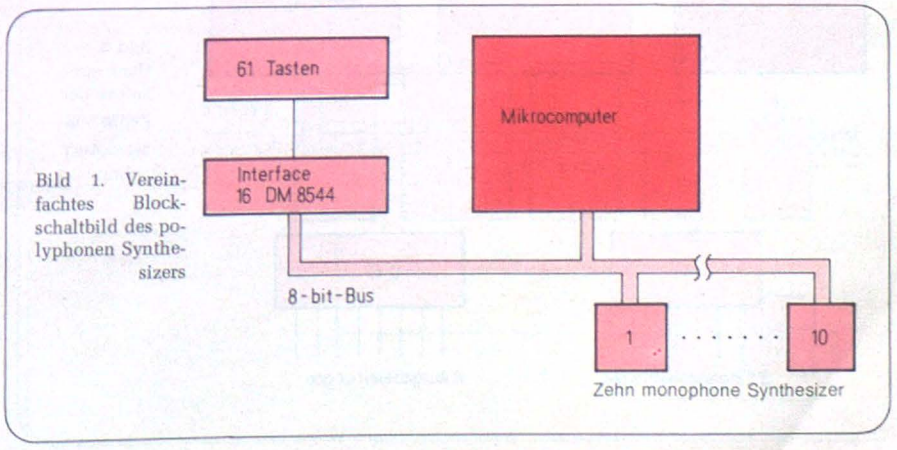


Bild 1. Vereinfachtes Blockschaltbild des polyphonen Synthesizers

sten zugleich wird ein Musiker kaum anschlagen. Die Aufgabe einer Ablaufsteuerung innerhalb eines polyphonen Synthesizers wird also darin bestehen, jeder gedrückten Taste einen freien Synthesizer zuzuweisen und ihn mit der richtigen Tonhöheninformation zu versorgen. Insbesondere ist es wichtig, nicht mehr angeschlagene Tasten schnell zu erkennen, da dann der Synthesizer, der vordem für diese Taste arbeitete, für eine beliebige andere Taste verfügbar wird.

Dazu wird die Stellung der 8 Tasten, die gleichzeitig über einen Eingangsport erfaßt werden, eingelesen und mit der Konfiguration der letzten Abfrage (die im Speicher an 8 RAM-Positionen MBn steht) über eine EX-OR-Verknüpfung verglichen. Das Ergebnis wird an einem der 8 RAM-Plätze △b abgelegt. Wenn keine Änderung erfolgte, liest der Prozessor den nächsten Eingangsport. Diese Parallelverarbeitung erlaubt die Verwendung eines relativ langsamen Prozessors, wie der ursprünglichen p-Kanal-SCMP-Version.

In der ersten Verarbeitungsphase (RESET-Phase), die der Abfrage folgt, wird zuerst nach Tasten gesucht, die nicht mehr gedrückt sind. Damit sollen soviel Synthesizer wie möglich frei gemacht werden. Die Erkennung einer nicht mehr angeschlagenen Taste erfolgt über eine UND-Verknüpfung zwischen der oben erwähnten EX-OR-Funktion und der Tastenstellung der vorangegangenen Abtastung.

Bevor die RESET-Anforderung verarbeitet wird, rettet der Prozessor die gegenwärtige Tastenstellung in den RAM-Bereich MBn durch Überschreiben des letztgültigen Wortes.

Liegt keine RESET-Anforderung vor, dann wird der nächste Eingangsport gelesen.

Aus Zeitgründen wird das Wort, welches die RESET-Anforderung enthält, auf einen Prioritätsencoder (CD 4532) gegeben, der augenblicklich das Problem der Mehrfachanforderung löst. Der SCMP-Mikroprozessor generiert jetzt zu der als nicht mehr angeschlagen erkannten Taste den richtigen Noten- und Oktavcode. Der Prioritätsencoder ist bei dem neueren SCMP-II-Prozessor in n-Kanaltechnologie nicht mehr nötig. Seine höhere Geschwindigkeit ermöglicht eine softwaremäßige Lösung des Problems der Mehrfachanforderung.

Nachdem jetzt das Steuerwort zusammengestellt wurde, fragt der Prozessor so lange die Peripherie ab, bis er einen Synthesizer mit identischem Code gefunden hat. Dieser hat ja bisher den Ton für die losgelassene Taste erzeugt und muß nun abgeschaltet werden. Die Abfrage einer gesteuerten Peripherie bedingt, daß die Peripherieports sowohl Eingang als auch Ausgang sein müssen. Dies ist recht aufwendig und damit auch entsprechend teurer. Man hilft sich, indem den Ausgangsports die gleiche Adresse wie einem RAM-Bereich zugewiesen wird, in dem dann eine Kopie der Peripherieinformation steht.

Wenn über diese Peripherieabfrage im RAM der zugeordnete Synthesizer gefunden ist, wird dessen MSB (Gate) auf "1" gesetzt, der Noten- und Oktavcode jedoch nicht geändert, damit das Ausklingen mit der richtigen Tonhöhe erfolgt.

Nachdem alle RESET-Anforderungen innerhalb dieses einen Eingangsports verarbeitet wurden, erfolgt die Abfrage des nächsten. Nach dem letzten Eingangsport wird dann die SET-Phase begonnen. In dieser Phase wird nach neu angeschlagenen Tasten gesucht.

Die SET-Phase arbeitet ausschließlich mit gespeicherten Daten. Tatsächlich ist die UND-Verknüpfung zwischen dem EX-OR in △n und der jetzigen Tastenstellung (in MBn) die SET-Abfrage, d. h. sie gibt an, welche Taste nach der vorletzten Abtastung angeschlagen wurde. Enthält △n nur Nullen, dann wird die nächste Speicherzelle abgefragt, bevor die UND-Verknüpfung erfolgt. Das gleiche geschieht, wenn das Ergebnis der UND-Verknüpfung Null ist.

Ist der Anschlag einer Taste erkannt, erzeugt der Prozessor den Noten- und Oktavcode.

Jetzt wird die Peripherie nach einem freien Synthesizer abgefragt (dessen MSB auf „1" ist). Nachdem einer erkannt ist, wird ihm der Steuercode und im MSB eine „0" übermittelt.

Andere mögliche SET-Anforderungen innerhalb einer △n-Speicherzelle werden verarbeitet, bevor die nächste Zelle untersucht wird. Nach der Behandlung der letzten Speicherzelle beginnt der Prozessor wieder von vorn.

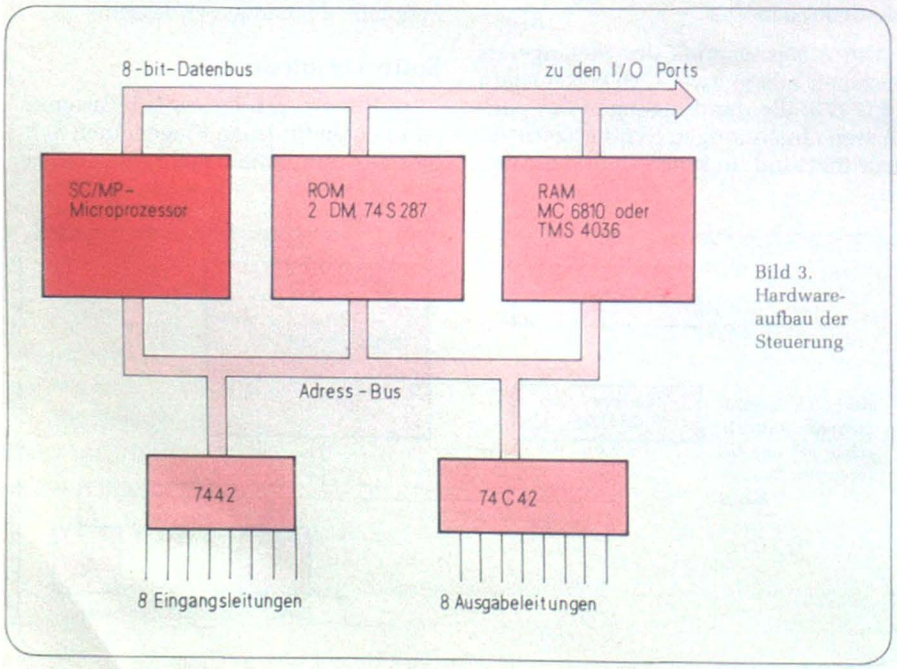


Bild 3. Hardwareaufbau der Steuerung

Softwarezusätze

Mit den beiden Steuereingängen des SCMP (die im Status Register mit den beiden LSB des Steuercodes übereinstimmen), läßt sich einfach eine Oktaventransponierung über 3 Oktaven erreichen. Der Softwareaufwand dafür ist minimal.

In die Abfrageschleife für die Peripherie (Schritte 4 bis 10) sind zusätzlich zwei Sicherungen eingebaut, erstens für den Fall, daß in der SET-Phase 11 angeschlagene Tasten erkannt wurden und zweitens falls der Mikroprozessor in der RESET-Phase, aus welchen Gründen auch immer, keine Übereinstimmung zwischen dem Steuerbyte und der Peripherie feststellt.

Dazu werden zwei RAM-Plätze benötigt, welche auf die zehn für die Peripherie belegten folgen. Bevor die Suchschleife startet, schreibt das Programm den Steuercode mit dem Gate auf „0" in den ersten dieser beiden Plätze (RESET-Sicherung) und einen beliebigen Noten- und Oktavcode mit dem Gate auf „1" in den zweiten Platz (SET-Sicherung). Die Peripherie wird dadurch

nicht beeinflußt, da es keine solchen mit entsprechender Adresse gibt.

Hardwareentwurf

Benötigt werden 28 RAM-Speicherplätze, 8 für MBn, 8 für △n, 12 für die Peripherie und die Sicherung. Es genügt daher ein 32 x 8-bit-RAM. Wegen der Verfügbarkeit wurde aber auf den MC 6810 von Motorola (128 x 8) zurückgegriffen, der damit aber nur zu einem Viertel ausgenutzt ist. Das gesamte Steuerprogramm, einschließlich der Notencodes und der Prioritätsencoderliste, findet in einem 256 x 8-bit-ROM Platz. Für den Prototyp eignet sich der EPROM 1702, während für die Produktion die Kombination von zwei PROMs DM 74 S 287 (je 256 x 4) vorteilhafter ist. Die Hardware umfaßt weiter noch: einen 7442 für die Auswahl der Eingangsports DM 8544 und einen 74 C 42 für die Adressierung der Ausgangsports, die aus je 2 74 C 175 bestehen. *Bild 3* zeigt den grundsätzlichen Aufbau des Systems, dessen Bauteilekosten um DM 50.– herum liegen dürften.

Eine einzelne Peripherieeinheit ist in *Bild 4* näher dargestellt: Der Ausgangsport steuert zwei Einhüllengeneratoren ADSR 1 und ADSR 2 mit dem Gatesignal und einen digital gesteuerten Oszillator mit dem restlichen 7 bit Code. Der Generator erzeugt Sägezahn, Rechteck und Impulse mit veränderbarem Tastverhältnis. Dessen Signale werden einem spannungsgesteuerten Resonanzfilter vierter Ordnung zugeführt, welches aus 4 OTA's (CA 3080) besteht [6]. Ein Signal vom Oszillator führt die Mittenfrequenz dieses Filters automatisch nach.

Der spannungsgesteuerte Verstärker ist eine Anordnung mit variablem Leitwert mit Vorentzerrung [7] und verwendet das Transistorarray CA 3086. Die Oszillatoren sämtlicher zehn Synthesizer werden gemeinsam von einem 550-kHz-Muttergenerator gesteuert; daher besitzt dieser polyphone Synthesizer gute Intervallreinheit und Absimmbarkeit.

Durch elektronische Beeinflussung von Mutteroszillator, Filter und Abschwächer kann Vibrato, WHA-WHA und Tremolo erzeugt werden. Die Hardware-Kosten für eine Peripherieeinheit dürften bei etwa 100 DM liegen.

Zusammenfassung

Das beschriebene Instrument ist ein polyphoner Synthesizer, dessen Klangfarbe für alle Töne ähnlich ist, wie es etwa auch beim Klavier ist. Der Zuordnungsalgorithmus, d.h. die Lösung der Frage, welcher Synthesizer für welche Tasten eingesetzt wird, enthält hier die Bildung einer Zeitpriorität nach Hovey und Seamans [3]. Ein großer Vorteil des Mikrocomputeransatzes ist die Flexibilität. Der SCMP von National Semiconductor wurde vornehmlich wegen seiner geringen Kosten ausgesucht. Die größte Verzögerungszeit zwischen Tastendruck und Ausgangssignal beträgt nur wenige Millisekunden und wird vom Spieler nicht bemerkt.

Abschließend soll angemerkt werden, daß das System mit einer beliebigen Anzahl von ein bis zehn Peripherieeinheiten betrieben werden kann.

Niedrige Anfangskosten durch 3 bis 4 Synthesizer bis zum vollen Ausbau geben eine weite Skala der Anwendungsmöglichkeiten.

Literatur

[1] Robert A. Moog: „Voltage Controlled Music Modules", J. Audio Eng. Soc. Vol. 13, Nr. 3, S. 200...206 (July 1965).

[2] D. Rossum: „Polyphonic Synthesizers", Electronotes Vol. 4, Nr. 32, S. 8 (1973).

[3] C. Hovey; D. Seamans: „A Polyphonic Keyboard Controller for a Voltage Controlled Music Synthesizer", J. Audio Eng. Soc. Vol. 23, Nr. 6, S. 459...465 (July) August 1975.

[4] Mark Goldstein: „8-voice Digital Keyboard Design Idea", Electronotes Vol. 8, Nr. 70, S. 18 (1976).

[5] Lee Ferguson: „A Polyphonic Music Synthesizer Utilizing Master-programmed Electronic Synthesis Modules for Each Key", AES Preprint Nr. 1110 (D-6) 54th Convention May 1976.

[6] Bernard Hutchins: „A four-pole Voltage Controlled Network", Electronotes Nr. 41, S. 7.

[7] Walter G. Jung: „IC-OpAmpCookbook", S. 251...253 (Howard W. Sams and Co. Inc., Indianapolis, 1974).

µC-Nachrichten

Noch 1979 will Intel eine neue 8-bit-CPU, voraussichtlich unter der Bezeichnung **8088,** herausbringen. Sie soll bus-kompatibel mit dem 8085 sein, aber einige Vorteile der 16-bit-CPU 8086 aufweisen. Außerdem plant Intel einen **Mathematik-Prozessor,** der eine deutlich schnellere Verarbeitung arithmetischer Probleme ermöglichen soll.

□

Allein Radio Shack (Tandy) will dieses Jahr rund 6000 Heimcomputer in **Japan** absetzen. Auch dort scheint nun also die Computer-Revolution der kleinen Systeme zu beginnen – den T 400 von Toshiba hatten wir ja schon kürzlich vorgestellt, und nun haben auch Casio, NEC, Fujitsu und Hitachi angekündigt, Heimcomputer auf den Markt zu bringen.

□

Die handlichen **Übersetzungs-Computer** – sie enthalten in ihrem taschenrechner-großen Gehäuse eine alphanumerische Tastatur und eine ebensolche Anzeige – entwickeln sich offenbar zu einem ebenso großen Verkaufsrenner wie bisher die Schachcomputer: Allein die Firma Lexicon plant für 1979, rund 200 000 solche elektronischen Wörterbücher zu produzieren. Ihr „Gedächtnis" beziehen die Geräte meist aus einschiebbaren ROM-Moduln; die Preise liegen in den USA bei etwa 200...300 Dollar.

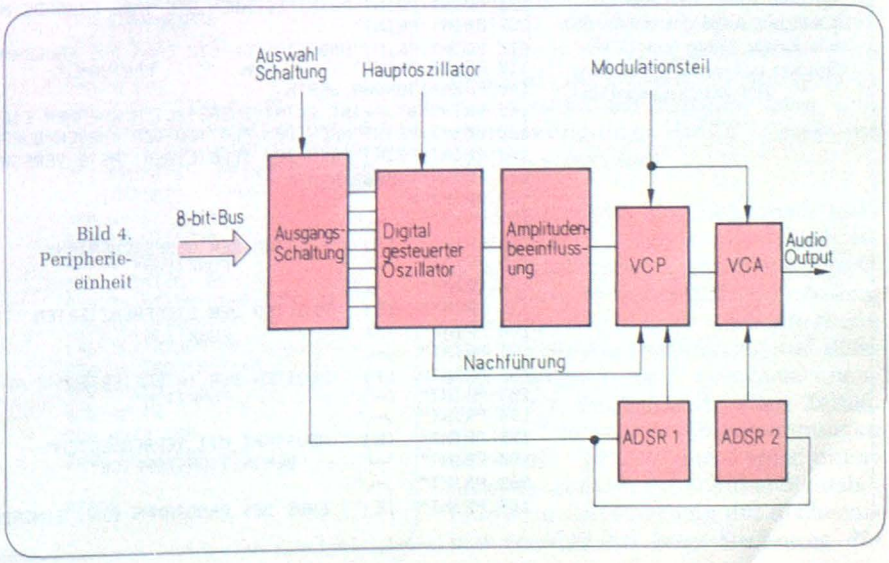


Bild 4. Peripherie-einheit

Peter-Mattias Oden

PET plottet Funktionen

Um kontinuierliche Funktionskurven darzustellen, ist der Mikrocomputer PET nur bedingt geeignet, da er einzelne Bildpunkte des Schirms nicht separat ansprechen kann, sondern immer nur Felder mit 8 x 8 Punkten. Mit einem kleinen Trick läßt sich trotzdem eine hinreichend gute Auflösung erreichen.

Durch die zahlreichen Grafiksymbole, die der PET erzeugen kann, hat er auch die Möglichkeit, die 8 waagerechten Linien dieser Felder einzeln anzusprechen. Dadurch kann man eine maximale Auflösung von 200 x 40 „Strichen" erreichen, was für einen Überblick über Eigenschaft und Form der dargestellten Funktion völlig ausreicht.

Das hier beschriebene Programm hat eine etwas geringere Auflösung. Der besseren Übersichtlichkeit wegen werden noch ein Koordinatensystem, die Nullinie der Funktion (falls sie innerhalb des Wertebereiches liegt), sowie der anzugebende Definitionsbereich und Wertebereich angezeigt.

Die darzustellende Funktion steht in Zeile 230 des Programms (*Bild*) und kann jederzeit neu definiert werden.

BASIC-Programm zum Plotten mathematischer Funktionen auf dem Bildschirm des PET 2001. Der Ausdruck entstand mit einem Centronics-779-R-Drucker mit einem speziellen PET-Zeichengenerator

```
1 I=PEEK(630):IFI/2>0THEN PRINT"ACHTUNG----"I"LESEFEHLER! BITTE NEU EINLESEN":NEW
2 PRINT "s":PRINT"GRAPHISCHE DARSTELLUNG VON FUNKTIONEN
3 PRINT:PRINT:PRINT:PRINT"COPYRIGHT    : PETER-M. ODEN
4 PRINT:PRINT"      DROSSELWEG 26"
5 PRINT:PRINT"    5042 ERFTSTADT-12"
6 FORN=1TO2000:NEXTN
10 DIMY(52):Y(0)=0
20 DIMZ(8):FORN=1TO8:READZ(N):NEXTN
25 DATA100,82,70,64,67,68,69,99
100 PRINT"s"
110 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
120 PRINT"SIND SIE MIT DER BEDIENUNG VERTRAUT?":PRINT:PRINT:PRINT"                (J/N)"
121 GETA$:IFA$=""GOTO121
122 IFA$="J"GOTO200
123 IFA$<>"N"GOTO10
124 PRINT"s":PRINT:PRINT:PRINT:PRINT:PRINT"DIESES PROGRAMM ERMOEGLICHT EINE GRAPHI-"
125 PRINT"SCHE DARSTELLUNG VON FUNKTIONEN AUF DEM ":PRINT"BILDSCHIRM IHRES"
126 PRINT:PRINT:PRINT:PRINT"ucccci ucccci ucccci"
127 :PRINT"bucih bu—k j—iu—k"
128 :PRINT"bj—kh bj—i   ||  "
129 :PRINT"bu—k bu—k   ||  "
130 :PRINT"b|    bj—i   ||  "
131 :PRINT"jk    j—k   jk  "
132 PRINT    :PRINT:PRINT:PRINT:PRINT:FORN=1TO3000:NEXTN
133 PRINT"♥":PRINT:PRINT:PRINT:PRINT"GEBEN SIE DIE GEWUENSCHTE FUNKTION  IN":PRINT:PRINT:PRINT
134 PRINT:PRINT:PRINT"  230  Y=FUNKTION (X)"
135 PRINT:PRINT:PRINT"BEISPIELSWEISE":PRINT:PRINT:PRINT"      230  Y=SIN(X)"
136 PRINT:PRINT:PRINT"BEENDEN SIE IHRE EINGABE MIT "
137 PRINT:PRINT"                RRETURN_"
138 PRINT:PRINT"UND STARTEN SIE DANN DAS PROGRAMM MIT"
139 PRINT:PRINT"       RUN        RRETURN_"
140 FORN=1TO9000:NEXTN
141 PRINT"♥":PRINT:PRINT"ZUSAETZLICH KOENNEN SIE DANN - UND AUCH"
142 PRINT:PRINT"NACH DEM PLOTTEN DER KURVEN DURCH DRUE-"
143 PRINT:PRINT"CKEN DER JEWEILIGEN TASTE VERSCHIEDENE":PRINT:PRINT"MOEGLICHKEITEN WAEHLEN.
144 FORN=1TO5000:NEXTN
150 PRINT"♥"
155 PRINT"   ⌒"
160 PRINT"  |F|  :PLOTTEN DER GESPEICHERTEN"
165 PRINT"   ⌣           FUNKTION"
170 PRINT"   ⌒"
175 PRINT"  |D|  :PLOTTEN DER DIFFERENZIERTEN"
180 PRINT"   ⌣           FUNKTION"
181 PRINT"   ⌒"
182 PRINT"  |I|  :PLOTTEN DER INTEGRIERTEN"
183 PRINT"   ⌣           FUNKTION"
184 PRINT"   ⌒"
185 PRINT"  |N|  :NEUSTART MIT VERAENDERTEM"
186 PRINT"   ⌣      DEFINITIONSBEREICH"
187 PRINT"   ⌒"
188 PRINT"  |E|  :ENDE DES PROGRAMMS BZW. EINGABE"
```

Nach Angabe des Bereiches, in dem die Funktion dargestellt werden soll, berechnet der PET die einzelnen Funktionswerte y(x) und sucht sich daraus den größten und den kleinsten heraus. Anschließend werden die Werte so umgerechnet, daß der PET weiß, welche Linie der 8x8-Matrix im Schirmbild er jeweils ansprechen muß.

Zusätzlich bietet das Programm die Möglichkeit, die einzelnen Kurven zu differenzieren, zu integrieren oder den darzustellenden Bereich zu verändern.

Zur Differentiation ist zu bemerken, daß, genau wie nach einer Integration, die ausgedruckten Wertebereiche mit Vorsicht zu genießen sind.
Bei jeder Integration entsteht ein konstanter Faktor, der sich bei weiteren Integrationen durch additive lineare oder quadratische Terme usw. bemerkbar macht.

Dem Fachmann mögen manche Teile dieses Programms als wenig elegant oder gar umständlich erscheinen. Das liegt daran, daß der Verfasser erst wenige Wochen mit dem PET gearbeitet hat und auch vorher keinerlei Erfahrungen mit BASIC hatte. Es wurde trotzdem konsequent versucht, mit den zur Verfügung stehenden Möglichkeiten des PET eine Anwendungsmöglichkeit aufzuzeigen, die außerhalb der üblichen Spiel-, Rechen- oder Sortierprogramme liegt.

Als Anregung noch einige interessante Funktionen:

Funktion	*von*	*bis*
y = SIN(x)	0	7
y = EXP(–x · x)	–2	2
y = x ↑ 3	–5	5
y = x · (x–2) · (x+1)	–1,5	2,5

```
100 PRINT"  ◡          EINER NEUEN FUNKTION"
110 FORN=1TO9000:NEXTN
120 PRINT"♥":END
130 PRINT"♥":PRINT:PRINT"BITTE NENNEN SIE MIR
NUN- DURCH KOMMA   GETRENNT -
140 GRENZEN DES"
150 PRINT"DEFINITIONSBEREICHES ; ":PRINT:PRINT
" ALSO X-MIN UND   X-MAX ."
160 INPUTA,B
170 C=B-A:IFC<=0GOTO200
180 D=C/37
190 F=-(1E38)
200 G=1E38
210 FORN=0TO51
220 X=A+N*D
230 Y=(COS(X))↑2
240 Y(N+1)=Y
250 IFN>37GOTO260
260 IFY(N+1)>FTHENF=Y(N+1)
270 IFY(N+1)<GTHENG=Y(N+1)
280 NEXTN
290 PRINT"♥":PRINT"BITTE GEBEN SIE NUN
300 GEWUENSCHTE DAR-STELLUNGSWEISE EIN"
310 PRINT"ALSO  F , D , I  ."
320 GETA$:IFA$=""THEN320
330 IFA$="E"GOTO1000
340 IFA$="D"GOTO500
350 IFA$="I"GOTO600
360 IFA$="F"GOTO700
370 IFA$="N"GOTO200
380 GOTO270
390 F=-(1E38)
400 G=1E38
410 FORN=1TO51
420 Y(N)=(Y(N+1)-Y(N))/D
430 IFN>38GOTO530
440 IFY(N)>FTHENF=Y(N)
450 IFY(N)<GTHENG=Y(N)
460 NEXTN
470 Y(52)=Y(51)
480 GOTO700
490 F=-(1E38):G=1E38
500 FORN=1TO51
510 Y(N+1)=Y(N)+Y(N+1)
520 IFN>38GOTO650
530 IFY(N)>FTHENF=Y(N)
540 IFY(N)<GTHENG=Y(N)
550 NEXTN
560 PRINT"♥"
570 FORN=0TO39
580 POKE(32808+N),99
590 NEXTN
600 FORN=0TO39
610 POKE(33728+N),100
620 NEXTN
630 FORN=0TO23
640 POKE(32808+N*40),101
650 NEXTN
660 FORN=0TO23
670 POKE(32847+N*40),103
680 NEXTN
690 POKE32808,79
700 POKE32847,80
710 POKE33728,76
720 POKE33767,122
730 POKE33738,76
740 POKE33748,76
750 POKE33758,76
760 POKE32818,79
770 POKE32828,79
780 POKE32838,79
790 E=F-G:IFE<.01THENG=G-
    (.01-E):F=F+(.01-E):E=F-G
800 H=SGN(G):I=SGN(F)
810 J=H+I:IFJ<>0GOTO792
820 K=(-G/E)*21:L=INT(K+.01):
    M=INT((K-L)*8+.02)+1
830 O=(L+1)*40:IFM<1THENM=1:
    IFM>8THENM=8
840 FORN=0TO39
845 P=N-O+33728
850 POKEP,Z(M)
860 NEXTN
870 POKE(33728-O),48
880 V=F*1000:V=INT(V):V=V/1000
890 W=G*1000:W=INT(W):W=W/1000
900 PRINT"S";:PRINTA"( X ("B;
    :PRINT"    "W"( Y ("V
910 Q=(E/170)*8
920 FORN=1TO38
930 R=(Y(N)-G)/Q
940 S=INT(R+.01)
945 T=INT((R-S)*8+.02) +1
946 IFT<1THENT=1
950 IFT>8THENT=8
960 U=33688+N-S*40
970 POKEU,Z(T)
980 NEXTN
990 GOTO 300
1000 PRINT"♥":END

READY.
```

Einige wichtige Variablen:

A = X-MIN
B = X-MAX
G = Y-MIN
F = Y-MAX
S = Zum Funktionswert gehörende Zeile
T = Zum Funktionswert gehörende Linie innerhalb der 8x8-Matrix

Die einzelnen Unterprogramme sind:

1-191	Einführung
200-260	Berechnen der Funktionswerte im gegebenen Bereich
270-360	Abfrage der gewünschten Darstellungsart
500-550	Berechnen der Funktionswerte der differenzierten Kurve
600-650	Berechnen der Funktionswerte der integrierten Kurve
700-794	Schreiben des Koordinatensystems, der Nullinie und des Definitions- und Wertebereiches
800-880	Zeichnen der Funktion ■

Gute Noten dank PET

Einige Schüler verschiedener Gymnasien in Baden verdanken es einer skurrilen Eigenwilligkeit des Mikrocomputers PET 2001, daß ihre Latein- und Mathematiknoten in den letzten beiden Zeugnissen um eine Kleinigkeit besser ausfielen, als es eigentlich verdient gewesen wäre. Möglicherweise – das läßt sich heute nicht mehr rekonstruieren – mag es in dem einen oder anderen Fall dank des Rechner-Fehlers noch zu einer knappen Versetzung gereicht haben, obgleich die Zeichen eigentlich auf „Sitzenbleiben" standen. Der folgende Beitrag geht auf die Hintergründe der „Affäre" ein und zeigt an einem konkreten Beispiel, unter welchen Bedingungen der PET Gnade vor Recht ergehen ließ.

Zeugnisnoten an baden-württembergischen Schulen werden nach einem komplizierten Verfahren ermittelt. Die Noten in schriftlichen Arbeiten, mündliche Leistungen sowie die Noten für anderes wie Heftführung und Referate werden je nach Kategorie unterschiedlich gewichtet. In alten Zeiten, als die Lehrer ihre Notenrechnungen allein mit Bleistift und Papier unternahmen, ordnete die Schulaufsichtsbehörde zur Erleichterung der Rechenarbeit an, daß bei jeder Rechnung die

zweite und jede folgende Stelle nach dem Komma wegzulassen sei. Ein Schnitt von 3,39 ist also wie 3,3 zu behandeln und nicht auf 3,4 zu runden, wie es beispielsweise Taschenrechner mit Festkomma-Einstellung tun.

Als nun im Frühjahr letzten Jahres ein aus Mannheim stammender Lehrer zu seinem und seiner Kollegen Nutzen ein kleines Notenrechnungsprogramm für den PET schrieb, da berücksichtigte er diese Stellenweglaß-Anweisung der Kultusbehörde durch ein Basic-Statement folgender Form:

SCHNITT = SCHNITT *10:
SCHNITT = INT (SCHNITT):
SCHNITT = SCHNITT/10

Also eine Prozedur mit folgenden Schritten: Multiplikationen mit zehn, Entfernung aller Nachkommastellen, Division durch zehn. Gewissenhaft vorgenommene Probeläufe des Programms mit dieser Anweisung ergaben einwandfreies Funktionieren. Das Programm wurde in der Folge noch gestrafft und verhalf dann, mehrfach kopiert, vielen Lehrern zu einer spürbaren Erleichterung beim mühsamen Geschäft des Notenrechnens.

Dann aber – im Januar 1979 – konnte ein gegenüber Computern offenbar recht mißtrauischer Altphilologe der Versuchung nicht widerstehen, einige PET-Ergebnisse stichprobenhaft nochmals „zu Fuß" durchzurechnen. Und siehe da: In einem Fall, in dem die Handrechnung einwandfrei 3,3 ergab – was in diesem Fall eine Zeugnis-Endnote 4 und damit die Nichtversetzung des Schülers bedeutete –, meldete der PET 3,2 als Ergebnis. Mit der Konsequenz „Jahresnote 3" und damit Versetzung in die nächste Klasse. Immer neues Nachrechnen ergab: Der PET hatte versucht, einen Schüler illegal zu begnadigen.

Die darob verständlicherweise aufs höchste beunruhigten PET-Freunde unter den Lehrern glaubten erst einmal an einen Programmierfehler. Als sich ein solcher nicht finden ließ, landete die Angelegenheit schließlich auf dem Schreibtisch des Autors. Ergebnis einer Versuchsserie: Kein Fehler im Programm, sondern ein tatsächlich im PET eingebauter „Sozialrabatt" für Notenrechnungen: Wirksam jedoch nur dann, wenn man komprimiert programmiert.

Das *Bild 1* zeigt die fraglichen Programmsegmente, in denen der PET seine widersprüchliche Natur offenbart. In einer ersten Fassung ist das Entfernen unzulässiger Stellen in Zeile 20 zusammengefaßt, in der Zweitfassung geschieht das gleiche – Zeile 50 – hintereinander. *Bild 2* zeigt, daß der Rechner mit gleichen Ausgangsdaten und gleichem, wenn auch anders geschriebenem Lösungsweg tatsächlich zu verschiedenen Ergebnissen kommt. Ursache? Die von Commodore Deutschland unterrichteten PET-Väter im fernen Amerika suchen noch danach. Bis man also Genaueres darüber hört, muß man wohl weiter davon ausgehen: Der PET hat ein Herz für schwache Schüler.

Hans-Georg Joepgen

Dezimal-Binär-Umwandlung mit dem HP-25

Mit dem hier vorgestellten Programm, das auf programmierbaren Taschenrechnern des UPN-Typs angewendet werden kann, lassen sich Dezimalzahlen bis zu $2^{10} = 1024$ in Binärzahlen umrechnen. Lädt man 0,1 in den Speicher 0, 1 in den Speicher 1 und 0 in den Speicher 2, so kann man nach Eingabe der Dezimalzahl und nach Abarbeiten des Programms in der *Tabelle* das Ergebnis als Folge von Nullen und Einsen ablesen. Die richtige Reihenfolge ergibt sich dabei allerdings, wenn man den Rechner auf den Kopf stellt. Die Kürze des Programms rechtfertigt es, dies in Kauf zu nehmen.

Prof. Dr.-Ing. E. Walloschke

Tabelle: HP-25-Programm

01	Enter	10	RCL 3
02	STO 3	11	2
03	2	12	:
04	:	13	INT
05	FRAC	14	STO 3
06	x = 0	15	x = 0
07	GTO 10	16	GTO 21
08	RCL 1	17	RCL 0
09	STO + 2	18	STO x 1
		19	R
		20	GTO 03
		21	RCL 2
		22	FIX 9
		23	R/S

```
LIST
  1 PRINTCHR$(147)
  3 PRINT:PRINT:PRINT:PRINT
  4 REM
  5 PRINT"     NOTEN-PARADOXON"
  6 PRINT"     ==============="
  7 REM
  8 REM       FASSUNG EINS
  9 REM       ------------
 10 S=66/20
 20 E=INT(10*S)/10
 25 PRINT:PRINT
 30 PRINT" * ENDNOTE NACH FASSUNG EINS:"
E
 40 S=66/20
 47 REM
 48 REM       FASSUNG ZWEI
 49 REM       ------------
 50 E=S*10:E=INT(E):E=E/10
 60 PRINT"    * ENDNOTE NACH FASSUNG ZWE
I:"E
 70 GOTO70
READY.
```

Bild 1. Hier die beiden Programmsegmente, auf die es ankommt: Zweimal die gleichen Ausgangsdaten, zweimal die gleiche Rechenvorschrift: Entfernung der zweiten und aller folgenden Nachkommastellen des Inhaltes der Variablen E (Endnote)

```
NOTEN-PARADOXON
===============

* ENDNOTE NACH FASSUNG EINS: 3.2

* ENDNOTE NACH FASSUNG ZWEI: 3.3
```

Bild 2. Hier offenbart der PET, daß er unter gewissen Umständen bei Notenberechnungen ein Herz für schwache Schüler zeigt: Das Ergebnis nach Fassung 1 ist um eine Zehntelnote besser. Zehntelnoten entscheiden bisweilen über Versetzungen

Hans Werner

Mikrocomputer

Eine kleine Einführung

16. Teil

Speicher: Grundsätzliche Unterschiede

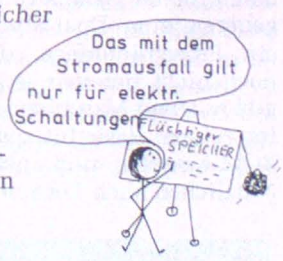
Speicher können Daten festhalten. Grob wird zwischen flüchtigen und nichtflüchtigen Ausführungsarten unterschieden.

Nichtflüchtige Speicher
...behalten ihren Inhalt auch bei Stromausfall. Aus ihnen können nur Daten ausgelesen werden. Der Informationsfluß ist einseitig.

Ausführungsformen: Tinte und Papier, Lochsteifen, mechanische und elektronische Lösungen. Wir wollen uns nur mit elektronischen Schaltungen beschäftigen.

In nichtflüchtigen Speichern stehen binäre Informationen auf Abruf bereit. Der Inhalt ist unveränderlich. Sie werden auch FESTWERT-SPEICHER genannt.

Flüchtige Speicher
...verlieren ihren Inhalt bei Stromausfall. Es können Daten ein- und ausgelesen werden. Der Informationsfluß kann in zwei Richtungen erfolgen. Ausführungsformen: Tafeln und Kreide, Flipflops usw.

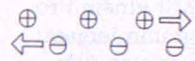


Die Informationen können jederzeit gespeichert und abgerufen werden. Flüchtige Speicher sind Schreib-Lesespeicher.

Speicher-Technologie

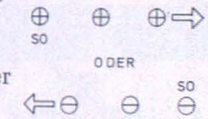
Beide Speicherarten können entweder aus bipolaren oder aus unipolaren Halbleiterelementen aufgebaut sein.

Bipolare Technik
Am Ladungstransport sind Elektronen und Löcher beteiligt. Die Steuerung erfolgt über den Minoritätsträgerstrom. Minoritätsträger sind die in der Minderheit vorhandenen beweglichen Ladungsträger (z. B. freie Elektronen in der p-Schicht).



Ausführungsarten: npn-Transistoren, pnp-Transistoren, Silizium- und Germaniumdioden, TTL- und I^2L-Schaltungen usw.

Unipolare Technik
Der Strom wird nur aus Majoritätsträgern gebildet (z. B. freie Elektronen in der n-Schicht). Es wird also nur eine Ladungsträgerart ausgenutzt, entweder Elektronen oder Löcher.

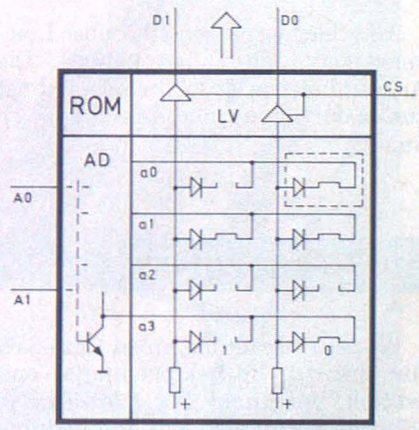


Ausführungsarten: Feldeffekt-Transistoren, MOSFETs, CMOS usw.

schiedenen Halbleitertypen nötig sind. Die folgenden Beispiele sollen gebräuchliche Begriffe erklären und sind nicht nach technologischen Gesichtspunkten geordnet.)

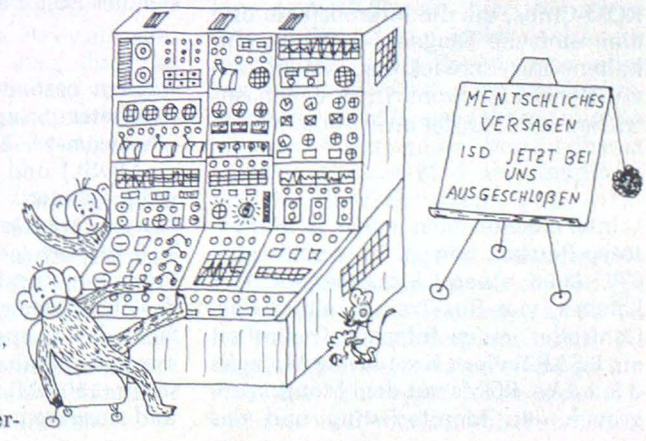
ROM

Das ROM ist ein nichtflüchtiger Speicher (Festwertspeicher). Die Daten sind fest vorhanden und können nur ausgelesen werden. Das *Bild* zeigt ein konventionelles ROM, dessen Spei-



	Bipolare Speicher	*Unipolare Speicher*
Bits pro mm²	meist wenig	viel
Zugriffszeit	kleiner 100 ns	ca. 500 ns
Stromverbrauch	groß	gering

In einem bipolaren Speicher sind z. B. 1024 Byte Speicherplätze auf 10 mm² untergebracht. Die Größe des gesamten Bausteins wird durch die Anschlüsse bestimmt.



(Dieser Abschnitt ist etwas kurz ausgefallen, weil für die weiteren Betrachtungen keine Kenntnisse über die ver-

cherzellen aus einer Diodenmatrix bestehen. Eine Kreuzung von Wortleitung a und Datenleitung D stellt eine Speicherzelle dar. Die Leitungen sind durch Dioden entkoppelt.

Jeder Adreßkombination A0, A1 ist durch den Adreßdekoder AD eine Wortleitung a zugeordnet. Ist z. B. A1 und A0 = 1, so wird a3 auf 0 V gezogen. Mit den Brücken kann eine 1 oder 0 programmiert werden.

A1	A0	
0	0	a0=0
0	1	a1=0
1	0	a2=0
1	1	a3=0

Nicht gebrückt
Die Zelle hat die Information 1, denn über den Widerstand liegt „+" an der entsprechenden Datenleitung.

Gebrückt
Information 0. Bei richtiger Adressierung ist die Datenleitung über die Diode auf 0 V geschaltet.

Beispiele
Mit A0 = 0 und A1 = 0 wird auch a0 = 0. D1 führt den Pegel „+", D2 liegt an 0 V (D0 =0, D1 =1).

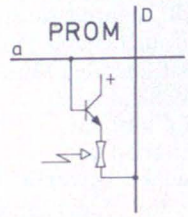
Alle Datenleitungen sind über Leseverstärker nach außen geführt. Die Auswahlleitung CS (Chip Select) dient zur Inaktivierung der Ausgänge (Tri-State).

Maskenprogrammierte ROMs
Während der Herstellung des Halbleiterspeichers wird mit einer sogenannten Maske ein Leiterbahnmuster mit den Informationen 0 oder 1 festgelegt.

Bei der Diodenmatrix enthält die Maske die Lage der Brücken. ROMs lohnen sich nur für große Serien und wenn keine Programmänderungen zu erwarten sind (Preis, Lieferzeit).

PROM

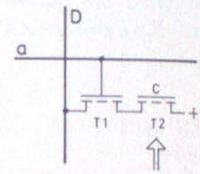
Programmierbare ROMs haben im Grundzustand auf allen Speicherplätzen eine 1. Mit einem Programmiergerät kann an den gewünschten Stellen die 1 in eine 0 verwandelt werden. Die Programmierung erfolgt z. B. durch Wegbrennen eines Widerstandes (dünner Nickel-Chrom-Film). Dazu wird über die Datenleitung ein im normalen Betrieb nicht vorkommender Strom (ca. 20 mA) geschickt.



EPROM

Erasable PROMs kann der Anwender selbst programmieren und wieder löschen (erase = ausradieren).

Sie sind mehrmals zu verwenden. Durch hohe negative Spannungen (50 V) wird eine negative Ladung auf das Gate von Transistor T2 gebracht, die erst nach Jahren wieder abfließen kann.



Ladung vorhanden: T2 ist niederohmig und T1 kann „+" auf die Datenleitung schalten (log. 1).

Keine Ladung: T2 ist hochohmig und T1 zieht die Datenleitung auf log. 0.

Fortsetzung folgt

µC-Nachrichten

Western Digital hat einen Chip-Satz für einen 16-bit-Mikrocomputer entwickelt, der direkt den **PASCAL**-Objektcode ausführen kann und dadurch rund fünfmal schneller als herkömmliche System-Software ist. Er trägt die Bezeichnung „Micro engine" und verwendet die an der Universität von San Diego geschaffene PASCAL-Version. Der Chip-Satz besteht aus einem Arithmetik-, einem Steuer- und drei ROM-Chips, die die Mikrobefehle und eine einfache Diagnose-Software enthalten. Ein Entwicklungssystem, das von Spezial Electronic in München vertrieben wird, kostet etwa 7000 DM.

□

Intel brachte einen neuen **16-bit-Prototyp-Bausatz** heraus. Er umfaßt eine CPU 8086, einen Clockgenerator, vier Latches, vier Bus-Treiber, einen Bus-Controller, einen Interrupt-Controller, ein USART, vier 1 K x 4-bit-RAMs, zwei 2 K x 8-bit-ROMs mit dem Monitorprogramm, ein Monitorlisting und das MCS-86-Benutzerhandbuch in englischer Sprache, allerdings keine Platine.

□

N.A.S. senkte die Preise für den bekannten Mikrocomputer **Nascom-1** (s. Heft 25/1978) und bietet jetzt passend dazu auch ein Bus-Buffer-Board, Speicher-Erweiterungs-Platinen, ein Tiny Basic in EPROMs und einen 2,7-K-Assembler/Editor auf Kassette an.

□

Einen besonders preiswerten **6802-Computer** bringt Eltec heraus. Der **„Eurocom-1"** kostet etwa 350 DM (+ MwSt.) und wird als fertiger Baustein geliefert. Er besitzt bereits ein Kansas-City-Kassetteninterface und ein LED-Display nebst Hex-Tastatur zur Programmeingabe. Interessant ist auch das als Zubehör lieferbare Video-Interface, das knapp 200 DM kostet. Zusammen mit einer einfachen ASCII-Tastatur (150 DM) ist bereits die Text-Ein- und Ausgabe möglich. Der Eurocom ist vorwiegend für die Anwendung als Hobbycomputer und für Lernzwecke gedacht.

□

Eine Reihe von Peripheriegeräten für den **„Sorcerer"**, auch „Sokrates" genannt, bietet Datamega nun an. Dazu gehören eine Dual-Mini-Floppy und ein Prozeß-Interface, das allerdings noch nicht lieferbar ist. An Software gibt es einen Maschinensprache-Monitor, einen Basic-Interpreter in einer ROM-Kassette und neben anderem Nützlichen auch Fortran IV.

□

Eine standardisierte Systemlösung für **Mikrocomputer-Steuerungen** z. B. in Waschmaschinen, Wäschetrocknern oder Geschirrspülern hat ITT geschaffen. Es handelt sich um ein einbaufertiges Modul mit einem Display, einer Tastatur, einer Netzausfall-Überbrükkung und natürlich der Zentraleinheit in Form des Mikrocomputers. Die frontseitige Abdeckung ist durch eine Folie von Spritzwasser-Einflüssen geschützt. Das System wird ab Mitte 1979 von der ITT Bauelemente Gruppe hergestellt.

Hans-Joachim Regge

Der Mikrocomputer AIM 65

Seit kurzer Zeit ist der Mikrocomputer AIM 65 von Rockwell zu haben. Er arbeitet mit der CPU 6502, besitzt einen eingebauten Drucker, eine 20stellige alphanumerische Anzeige und eine ASCII-Tastatur. Sein Preis, unter 1000 DM, macht ihn auch für Hobby-Anwendungen interessant.

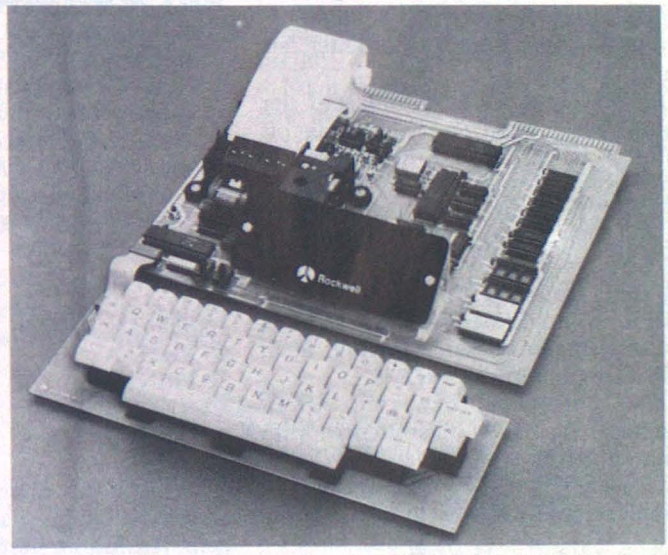
Wer sich einen Mikrocomputer anschaffen will, sollte sich zuvor überlegen, was er mit dem Gerät vorhat bzw. welche Möglichkeiten er sich offenhalten will: Sollen vorhandene oder zu kaufende Programme benutzt werden, so tut's fast jeder, vorausgesetzt, die Programme sind ihm auf den Leib geschrieben. Oft werden jedoch nur fertige Spielprogramme einige Male der staunenden Verwandtschaft vorgeführt, entlocken ein ehrfürchtiges „Aah", und dann kommt das Gerät wieder in den Karton, „weil man ja doch nicht viel damit anfangen kann", wie es heißt. Für den ehemals stolzen Besitzer kam das Erwachen, als er selbst eigene Programme erstellen wollte: Das Entwanzen (debugging), also das Finden von Fehlern in Programmentwürfen, gestaltet sich weitaus schwieriger als angenommen, weil die Unterstützung durch ein ausgefeiltes Monitorprogramm fehlt. Ein Beispiel (es ist etwas unfair, denn der KIM-1 kostet kaum die Hälfte des AIM 65): Wer beim KIM-1 den Inhalt der Register PC, PS, A, X, Y und des Stackpointers wissen will, muß 24 Tasten betätigen und sich zwischendurch noch die Werte notieren; bei vielen anderen Hobbycomputern ist es ähnlich (beim Durchfahren eines Programms in Einzelschritten sind diese Werte zur Erkennung von Fehlern oft nach jedem Einzelschritt erforderlich).

Beim AIM 65 *(Bild)* genügt ein Druck auf die Taste R, und die Registerinhalte werden mit Überschrift fein säuberlich nebeneinander auf dem eingebauten Drucker ausgedruckt. Die Begeisterung für dieses Gerät steigt, wenn nach dem Drücken der Taste I direkt in Assemblersprache (mnemonisch) programmiert werden kann und der Computer bei illegalen Befehlen, falschen Adressierungsarten oder bei unzulässigen Sprüngen den inneren Zeigefinger erhebt, „Error" ausdruckt und den richtigen Wert verlangt. Bei relativen Sprüngen übernimmt das Gerät selbst die Errechnung der lästigen Hex-Zahl und setzt sie ein, wenn man ihm das Sprungziel angibt. Auch das „verkehrte" Eingeben von Adressen (niederwertiges Byte zuerst, höherwertiges danach) fällt fort.

Ein Disassembler ist – wohlgemerkt alles in der „Sparausführung" mit 8-K-Monitor – auch vorhanden, erkennt Fehler und leistet gute Dienste bei der Aufschlüsselung von fremden Programmen. Fast 40 weitere Kurzkommandos, bestehend aus max. zwei Tastendrücken, lassen die Zeit beim Programmieren weiter schrumpfen.

Ein Cursor erlaubt die Korrektur von gerade verkehrt eingegebenen Werten. Der ebenfalls in der Grundausstattung bereits enthaltene Text-Editor speichert Texte, druckt sie als Block oder zeilenweise wieder aus, erlaubt die zeilenweise Ausbesserung und sucht den Text auch noch nach Stichworten durch. (Ein ähnliches Programm erschien in FUNKSCHAU 1978, Heft 24, für den KIM und nahm immerhin fast eine ganze Seite ein, hier ist's schon drin...) Bei einigen Hobbycomputern bereitet schon die Speicherung eines nur kleinen Textes arges Kopfzerbrechen. Die 38, teilweise sehr gut auch in eigenen Programmen verwendbaren Unterprogramme sind im Handbuch ausführlich beschrieben.

Als Display werden rotleuchtende LED-Anzeigebausteine verwendet; die Anzeige ist knapp 5 mm hoch und auch bei heller Beleuchtung gut lesbar. Einige Zeichen wirken zwar etwas gekünstelt, aber ein Fragezeichen braucht ja nicht unbedingt einen Punkt zu haben.



Einplatinen-Computer mit ASCII-Tastatur, alphanumerischer Anzeige und Thermodrukker: der AIM 65 von Rockwell. Der PC-100 von Siemens besitzt identische Eigenschaften und Monitorprogramme

Drucker mit Kompromissen

Versuche zeigten, daß der vom AIM 65 gelieferte Ausdruck wegen mangelnden Kontrastes kaum für eine Reproduktion in der FUNKSCHAU geeignet ist. Die Druckqualität ist bei Verwendung von üblichem Thermopapier nicht einmal für saubere Fotokopien geeignet. Aus diesem Grunde verzichteten wir hier auf die Wiedergabe von typischen AIM-65-Ausdrucken.

Die erwähnten Unterprogramme sind übrigens nicht identisch mit denen des KIM; es wird also kein KIM-Programm auf dem AIM laufen, welches auf Unterprogramme zurückgreift, was nun einmal fast alle tun. Als Ausgabebaustein wird im AIM 65 das IC 6522 verwendet. In diesem befinden sich neben zwei Datenregistern A und B, von denen jedes Bit herausgeführt ist und getrennt als Ein- und Ausgabeleitung einsetzbar ist, ein Schieberegister, zwei voneinander unabhängige Timer für die Erzeugung von 1...65 000 µs langen Impulsen ohne Programmschleife.

Leider ist der KIM-Timer völlig anders aufgebaut, und auch deshalb läuft kein KIM-Programm ohne Änderungen auf dem AIM, wenn es den Timer verwendet, z. B. das Baudot-Ausgabeprogramm aus dem Heft 1/79.

Wer schon einmal neben einem lärmenden Fernschreiber einige Zeit ausharren mußte, wird den eingebauten kleinen Thermodrucker schätzen. Er ersetzt – insbesondere bei der Programmherstellung – in wohl fast allen Fällen die große Maschine und arbeitet fast lautlos.

Erweiterungen sind bereits auf der Platine vorgesehen, was sonst leider oft vermißt wird: Die 1-K-RAM-Ausführung kann durch das Einstecken von 6 RAMs (2114) auf 4 K RAM erweitert werden. Fassungen sind auch noch für 3 ROMs à 4 KByte vorhanden. In (hoffentlich) kurzer Zeit sollen Basic und Assembler in ROMs lieferbar sein. In die Fassungen passen natürlich auch selbstprogrammierte PROMs.

Ein Videozusatz bzw. ein ASCII-Terminal und zwei Kassettenrecorder können selbstverständlich auch angeschlossen werden. Es ist beruhigend, daß das bewährte, oft schon preiswerte KIM-Zubehör paßt. Die mitgelieferten Unterlagen sind immerhin 7 cm dick (noch englisch) und ermöglichen bereits nach kurzer Einarbeitung die Erstellung von kleinen Programmen.

Zusammenfassend kann wohl gesagt werden, daß der AIM 65 der in seiner Preisklasse wohl leistungsfähigste Hobbycomputer auf dem Markt ist. Wer selbst Programmieren lernen möchte und auch später sein Gerät weiternutzen will, findet viele Hilfsmittel in der Monitor-Software, auf die Besitzer anderer Mikrocomputer, die ein Monitorprogramm von 1...2 KByte besitzen, neidvoll blicken werden.

Auch nach einer ersten Sturm- und Drangzeit wird der AIM nicht, wie so manches Videospiel, im Kasten bleiben, sondern er ist als Arbeitspferd einsetzbar.

Noch ein Tip: Wem die englische Sprache nicht so liegt, kann das deutschsprachig lieferbare Programmhandbuch des KIM verwenden – es ist dasselbe.

Nun, etwas ist am AIM 65 natürlich auch auszusetzen: Eine Demonstrationskassette wird auch hier leider nicht mitgeliefert. Manch einer hätte gern für einige ausgereifte Programme, von denen man mitunter viel abschreiben kann, noch etwas mehr ausgegeben. Im Anhang des Handbuches befindet sich nur ein – allerdings recht umfangreiches – Programm zur Überwachung der 16 Eingangsleitungen. Bei einer Änderung erfolgt ein Ausdruck der Uhrzeit sowie eines vorgegebenen Textes und einer Pegelangabe.

Kompatibilität mit dem KIM-1: Nicht vorhanden

Zwar verwendet der AIM 65 den gleichen Mikroprozessor wie der bekannte KIM-1, nämlich den 6502; KIM-Programme laufen aber leider nicht ohne weiteres auf dem AIM 65. Der Grund hierfür liegt nicht nur in den völlig anderen Unterprogrammen der Monitor-ROMs, sondern auch in der unterschiedlichen Struktur der Timer in KIM und AIM. Auch das Format, mit dem Daten auf einem Kassettenrecorder aufgezeichnet werden, ist nur begrenzt kompatibel: Das beim KIM recht verbreitete Hypertape-Format (800 Bd) wird – im Gegensatz zum KIM-Normalformat mit 135 Bd – vom AIM 65 nicht gelesen.

TRS-80 – ein Computer mit kleinem Fehler

Unbekannte Fehler in der ROM- oder RAM-Software eines Computers führen zu unerkannten oder unerklärlichen Fehlern im Ergebnis. Oft bleiben diese Fehler selbst dem Hersteller verborgen. Hier ein Beispiel: Dem Mikrocomputer TRS-80 von Tandy wird nach dem Einschalten folgendes BASIC-Programm eingegeben.

```
10 FOR I = 0 TO 3
20 READ A
30 PRINT A
40 NEXT I
50 DATA 1, 2, 3, 4
```

Die Analyse dieses einfachen Programmes ergibt, daß der Computer beim Ausführen die Ziffern 1 2 3 4 auf dem Bildschirm anzeigen wird. Tatsächlich erscheint aber viermal die Ziffer 1.

Das dem TRS-80 User's Manual (Level I) entnommene obige Programmbeispiel wird also falsch ausgeführt. Beim ordnungsgemäßen Arbeiten des Programms müßte, durch einen internen Zähler gesteuert, für I = 0 das 1. Element der Datenzeile der Variablen A zugeordnet werden, für I = 1 das 2. Element usw. Das sollte zum Ausdruck der Ziffern 1 2 3 4 führen.

Beim TRS-80 funktioniert dieser Zähler nicht. Das ist manchmal ärgerlich, wenn auf Grund dieser Tatsache einige korrekte, aber umfangreichere BASIC-Programme ähnlichen Typs falsch arbeiten. Der hier beschriebene Fehler erledigt sich aber oft unbemerkt von selbst, wenn der Computer zuvor ein längeres Programm ohne READ-Anweisungen ausführt.

Was aber ist zu tun, wenn sich ein Programm mit READ-Anweisungen im Arbeitsspeicher befindet und der interne Zähler nicht arbeiten will? Einen Ausweg bietet das Kommando:

POKE 16553,255

Nach einmaliger Eingabe dieses Befehles arbeitet der interne Zähler ordnungsgemäß. *Wolfgang Neske*

Maskenfehler im 6502

Wie Dr. Karl Meinzer bei der Entwicklung eines IPS-Interpreters feststellte, haben die Mikroprozessoren vom Typ 6502 aller Hersteller (!) einen Maskenfehler, der bisher offenbar niemandem aufgefallen war.

Er führt dazu, daß ein indirekter Sprungbefehl nicht korrekt ausgeführt wird, wenn zwischen dem höher- und dem niederwertigen Adreßteil ein „Page Crossing" stattfindet. Lautet der Befehl z. B. 6C FF 02, so wird die Sprungadresse nicht aus den Zellen 02FF und 0300 geholt, sondern fälschlich aus den Zellen 02FF und 0200, was zu einem völlig falschen Programmverhalten führt, das man sich zunächst nicht erklären kann.

Man muß bei indirekten Sprungbefehlen also darauf achten, daß das dem Operationscode 6C folgende Byte nicht FF ist. Fe.

Hans Vollmer

Dateneingabe mit Hexadezimal-Tastatur

Schaltung und Monitorprogramm dieses Beitrages dienten als Erweiterung des in der FUNKSCHAU 1977 unter dem Titel „Praxis mit Mikroprozessoren" vorgestellten Mikrocomputers. Es wird damit ermöglicht, ein Programm über eine Tastatur im Maschinencode in den Mikrocomputer einzugeben.

Notwendige Hardware

Da der Mikroprozessor 8080 immer acht bit lange Befehle verarbeitet, muß vor der Eingabe der Daten in den Speicher auch sichergestellt sein, daß ein acht bit langes Datenwort zur Verfügung steht. Möchte man diese Bits mit einem Tastendruck zur Verfügung stellen, bräuchte man 256 Tasten. Dies würde unübersichtlich und teuer werden. Also muß man durch sequentielles Drücken von zwei Tasten, welche im Hex-Code (4 bit) decodiert sind, die 8 bit zur Verfügung stellen (*Bild 1*).

Zur Decodierung dient eine Diodenmatrix, welche gleichzeitig noch ein Strobe als fünftes Bit enthält. Mit diesem Strobe-Signal wird die Entprellung der Tasten und Abspeicherung in Zwischenspeicher ermöglicht. Die Entprellung der Tasten geschieht dadurch, daß die Daten erst nach Ablauf einer Zeit (Prellzeit) in den Zwischenspeicher übernommen werden. Diese Zwischenspeicherung wird mit Hilfe von zwei monostabilen Kippstufen erreicht, wobei die erste vom Strobe angestoßen wird und die Prellzeit der Tasten überbrückt, und die zweite, angestoßen nach Ablauf der ersten, einen Schreibeingang des Zwischenspeichers freigibt.

Als Zwischenspeicher wurden acht D-Flipflops (74100) gewählt. Dieses IC hat zwei Takteingänge, die immer gemeinsam auf vier der acht Flipflops wirken. Damit ist es möglich, immer vier bit auf einmal abzuspeichern. Die Umschaltung, welche vier bit gerade eingeschrieben werden sollen, wird durch ein R-S-Flipflop (7474), das nach Ablauf der zweiten Zeitstufe umgeschaltet wird, bestimmt. Das RC-Glied am Setzeingang des Flipflops (7474) dient dazu, daß nach dem Einschalten der Betriebsspannung der Q-Ausgang auf H-Potential geschaltet wird, und dadurch nach Drücken der ersten Taste auch die ersten vier bit eingeschrieben werden.

Die 7-Segment-Anzeigen am Ausgang des Zwischenspeichers dienen zur Kontrolle der gerade eingegebenen Daten, können aber auch durch acht Leuchtdioden ersetzt werden. Die drei LED ADR, LOAD und RUN dienen zur Anzeige, in welchem Teil des Betriebsprogrammes sich der Mikrocomputer gerade befindet. Die Taste STO wurde über ein Flipflop entprellt, was bei den Tasten LOAD und RUN nicht erforderlich sind.

Die Bezeichnungen PA 0...7 und PC 0...6 sind identisch mit den gleichnamigen Pins des Ein/Ausgabebausteins 8255, welcher sich auf dem Ausgabemodul des Mikrocomputers befindet, und werden direkt miteinander verbunden.

Mit den Tasten 0...F können nun alle 256 mögliche Befehle, die der Mikroprozessor 8080 verarbeiten kann, in den Zwischenspeicher eingegeben werden, und erscheinen zur Kontrolle auf den beiden 7-Segment-Anzeigen. Die Übernahme der Daten in den Speicher des Mikrocomputers erfolgt erst nach Drücken der Taste STO, so daß eine Korrektur noch möglich ist. Mit den Tasten RUN und LOAD wird gewählt, ob ein Programm eingegeben werden soll (LOAD), oder ein bereits eingegebenes Programm abgearbeitet werden soll (RUN).

Nach dem Einschalten des Computers leuchtet die LED „ADR" auf. Der Computer signalisiert damit, daß die Anfangsadresse des einzugebenden oder zu verarbeitenden Programmes eingegeben werden soll. Da der Adreßbereich des Mikroprozessors 16 bit umfaßt, müssen zweimal 8 bit eingegeben werden. Hierbei wird wie folgt vorgegangen:

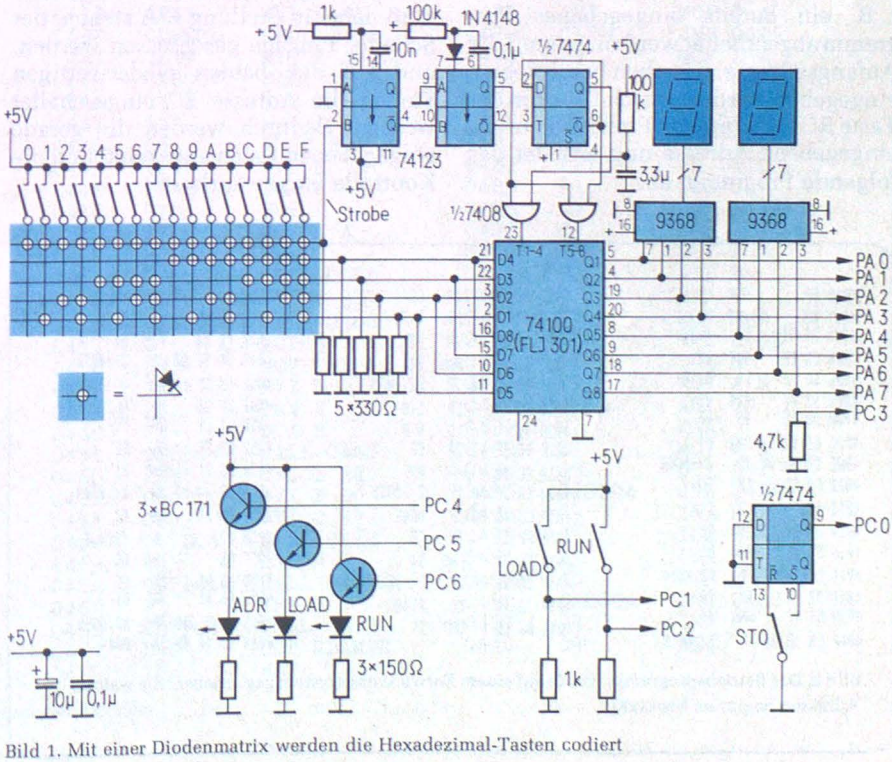


Bild 1. Mit einer Diodenmatrix werden die Hexadezimal-Tasten codiert

Zuerst werden die zwei niederwertigen Stellen der Adresse eingetippt. Sie erscheinen auf den beiden 7-Segment-Anzeigen. Durch Drücken der Taste STO wird dieses Byte in einem Register abgespeichert. Nun werden die beiden höherwertigen Stellen der Adresse eingegeben, und nochmals die Taste STO gedrückt. Danach muß gewählt werden, ob ein Programm eingegeben oder ein bereits eingegebenes Programm abgearbeitet werden soll. Durch Betätigen der Taste LOAD springt der Prozessor in den Teil des Betriebsprogrammes, der es ermöglicht, ein Programm ab der eingegebenen Anfangsadresse einzuprogrammieren und signalisiert dies durch Aufleuchten der LED LOAD.

Nun kann also ein selbst erstelltes Programm eingegeben werden. Es muß darauf geachtet werden, daß nach der Eingabe jedes Befehls immer die Taste STO gedrückt wird. Die Eingabe der Adresse braucht nur einmal zu erfolgen, da die Register, in denen die Adresse abgespeichert wurde, nach jeder Betätigung der Taste STO um eins erhöht werden.

Nach Eingabe eines Programmes ist es durch Betätigen der Taste RUN möglich, den Prozessor auf die Anfangsadresse des gerade eingegebenen Programms springen zu lassen. Der Prozessor signalisiert dies durch Aufleuchten der LED „RUN". Ein Rücksprung in den Adreßbereich des Betriebsprogramms ist nur durch Reset möglich. Soll nun z. B. ein bereits eingegebenes Programm abgearbeitet werden, so muß die Anfangsadresse, wie oben beschrieben, eingegeben werden. Durch Drücken der Taste RUN springt der Prozessor auf die eingegebene Adresse und arbeitet das folgende Programm ab.

Es ist auch möglich, eine Korrektur oder Änderung einer bestimmten Speicherzelle vorzunehmen. Hierbei wird die Adresse der Speicherzelle als Anfangsadresse eingegeben, die neue Information eingetippt und die Taste LOAD gedrückt. Allerdings muß zur Abarbeitung des Programmes die Anfangsadresse neu eingegeben werden, falls sie mit der Korrekturadresse nicht übereinstimmt.

Das Betriebsprogramm (*Bild 2*) ist so ausgelegt, daß der erste Ein/Ausgabebaustein 8255 des Ein/Ausgabemoduls angesprochen wird. Soll ein anderer Baustein verwendet werden, so muß das Betriebsprogramm auf den Adressen, auf denen eine Ein- bzw. Ausgabe stattfindet, entsprechend geändert werden. Der Kanal B des Ein/Ausgabebausteins, welcher noch unbenutzt ist, wurde als Ausgabekanal programmiert. Sollte er in einem späteren Programm benutzt werden und als Eingabekanal dienen, so kann der Ein/Ausgabebaustein jederzeit umprogrammiert werden. Die LED „RUN", die anzeigt, daß der Prozessor ein Programm abarbeitet, erlischt nach der Umprogrammierung des E/A-Bausteines. Ist jedoch von Anfang an bekannt, daß der Kanal B als Eingabekanal dienen soll, so können auf der Adresse 0003 des Betriebsprogrammes statt der Daten 91 die Daten 93 eingeschrieben werden.

Bei der Eingabe der Daten leistet die Zusatzschaltung zur Anzeige der Daten von einer vorgewählten Adresse gute Dienste. Der Schalter E/A-Speicher muß dabei in Stellung E/A stehen, der Schalter Eingabe geschlossen werden, und auf den beiden niederwertigen Stellen die Adresse EC eingeschaltet werden. Dadurch werden die gerade eingegebenen Daten angezeigt, und die Kontrolle ist gewährleistet.

```
4000 00     NOP
4001 00     NOP
4002 3E 91  LD   A,91
4004 D3 EF  OUT  EF
4006 3E 10  LD   A,10
4008 D3 EE  OUT  EE
400A DB EE  IN   EE
400C E6 01  AND  01
400E CA 0A 00 JP  Z/,000A
4011 DB EC  IN   EC
4013 4F     LD   C,A
4014 DB EE  IN   EE
4016 E6 01  AND  01
4018 C2 14 00 JP  NZ,0014
401B DB EE  IN   EE
401D E6 01  AND  01
401F CA 1B 00 JP  Z/,001B
4022 DB EC  IN   EC
4024 47     LD   B,A
4025 DB EE  IN   EE
4027 E6 01  AND  01
4029 C2 25 00 JP  NZ,0025
402C 69     LD   L,C
402D 60     LD   H,B
402E DB EE  IN   EE
4030 E6 02  AND  02
4032 CA 38 00 JP  Z/,0038
4035 C3 4E 00 JP  004E
4038 DB EE  IN   EE
403A E6 04  AND  04
403C CA 2E 00 JP  Z/,002E
403F 3E 40  LD   A,40
4041 D3 EE  OUT  EE
4043 E9     JP   (HL)
4044 DB EE  IN   EE
4046 E6 04  AND  04
4048 CA 52 00 JP  Z/,0052
404B C3 3F 00 JP  003F
404E 3E 20  LD   A,20
4050 D3 EE  OUT  EE
4052 DB EE  IN   EE
4054 E6 01  AND  01
4056 CA 44 00 JP  Z/,0044
4059 DB EC  IN   EC
405B 02     LD   (BC),A
405C 03     INC  BC
405D DB EE  IN   EE
405F E6 01  AND  01
4061 C2 5D 00 JP  NZ,005D
4064 C3 44 00 JP  0044
```

Bild 2. Das Betriebsprogramm wurde auf einem Entwicklungssystem geschrieben; die wahren Adressen beginnen bei 0000

Der Ausdruck des Betriebsprogrammes entstammt einem Entwicklungsplatz, auf dem das Programm in ein EPROM geladen wurde. Aus diesem Grunde beginnt das Programm mit der Adresse 4000, was aber tatsächlich der Adresse 0000 entspricht.

SCMP macht Musik

Das hier vorgestellte Programm (*Bild*) erzeugt eine zufällige Folge von Tönen und wurde ursprünglich für den Mikrocomputer MK-14 (Fa. Eltec) geschrieben. Es läßt sich aber ohne weiteres auch für andere SCMP-Systeme benutzen, da die MK-14-Monitorprogramme nicht verwendet werden. Lediglich eine Änderung der absoluten Adressen kann erforderlich werden.

```
0F12  A802  ILD
0F14  C024  LD
0F16  1E    RR
0F17  1E    RR
0F18  1E    RR
0F19  1E    RR
0F1A  1C    SR
0F1B  1C    SR
0F1C  1C    SR
0F1D  1C    SR
0F1E  1C    SR
0F1F  08    NOP
0F20  C80B  ST
0F22  C80E  ST
0F24  C460  LDI
0F26  C813  ST
0F28  C401  LDI
0F2A  07    CAS
0F2B  8F00  DLY
0F2D  C4C0  LDI
0F2F  07    CAS
0F30  8F00  DLY
0F32  B807  DLD
0F34  C005  LD
0F36  98DA  JZ
0F38  90EF  JMP
```

Programm zum Erzeugen von Zufallsmusik mit dem SCMP

Das Programm baut darauf auf, daß nach dem Einschalten des Mikrocomputers in den RAM-Zellen rein zufällige Werte stehen. Es holt sich Zelle für Zelle, formt den Inhalt in eine spielbare Tonhöhe um (Zeilen 0F16...01FE) und gibt den Ton am Flag-0-Anschluß des Mikroprozessors aus. Die Dauer der Töne wird durch den Wert in der Zelle 0F25 festgelegt. Ersetzt man den Befehl an der Adresse 0F1F (No Operation) durch 1C (Shift Right), so verschiebt sich die Tonhöhe etwas nach oben.

Um den Mikroprozessor-Flag-Ausgang vor Überlastung zu schützen, ist es sinnvoll, eine Darlington-Transistor-Treiberschaltung zu verwenden, um den Lautsprecher anzusteuern.

Thomas Obermair

Arno Ruff

Immerwährender Kalender und Biorhythmus

Der folgende Beitrag befaßt sich mit zwei Mikrocomputerprogrammen, die beide von der gleichen mathematischen Formel abgeleitet wurden. Die Programme sind speziell für den ITT-Mikroprozessor-Experimenter geschrieben, der als CPU den 8080-Chip enthält.

Da der ITT-Experimenter nur $^1/_4$ KByte RAM für das Anwenderprogramm aufweist, mußten die Berechnungen auf die Zeit vom 1.3.1900 bis 31.12.1999 beschränkt werden. Steht ein größerer Speicher zur Verfügung, kann die Berechnung selbstverständlich für einen größeren Zeitraum erfolgen. Die Formel hierfür lautet dann (1):

$N = (J \times 365{,}25) + (M \times 30{,}6) + T - K$
$K = 0$ vom 1.3.1700 bis zum 28.2.1800
$K = 1$ vom 1.3.1800 bis zum 28.2.1900
$K = 2$ vom 1.3.1900 bis zum 28.2.2100

Für die in diesem Beitrag beschriebenen Programme wurde folgende, vereinfachte Formel angewandt (2):

$N = (J \times 365{,}25) + (M \times 30{,}6) + T$

Vor der Berechnung sind bei beiden Formeln folgende Korrekturen vorzunehmen: Fällt das Datum auf Januar oder Februar, so ist zum Monat 13 zu addieren und beim Jahr 1 zu subtrahieren. Fällt das Datum auf März bis Dezember, wird lediglich 1 zum Monat addiert.

Beispiel 1: 2.1.1912. Hier fällt das Datum auf den Januar, das Ausgangsdatum für die Formel (1), (2) lautet also dann kurioserweise 2.14.1911.

Beispiel 2: 23.4.1943. Für April ist hier nur 1 zum Monat zu addieren. Das Berechnungsdatum ist dann der 23.5.1943.

Bei allen Multiplikationen fallen die Nachkomma-Stellen weg, erst dann wird addiert. Für beide Formeln gilt ebenfalls: Die Zahl N wird anschließend durch 7 dividiert. Der verbleibende Rest bezeichnet den Wert W, der den Code für den Wochentag darstellt:

W = 0: Sonntag
W = 1: Montag
W = 2: Dienstag
W = 3: Mittwoch
W = 4: Donnerstag
W = 5: Freitag
W = 6: Samstag/Sonnabend

Für den Anwender ist es wichtig zu wissen, daß die Programme sämtliche Korrekturen automatisch anbringen und die Berechnungen auch in der vorstehend beschriebenen Form durchführen. Es sind also lediglich die gewünschten Daten einzugeben, auf der Anzeige erscheint dann die Codezahl W, bzw. der jeweilige Anfangswert für die Bio-Zyklen.

Für die Jahreszahl werden nur die beiden letzten Stellen eingegeben, also statt 1978 z. B. nur „78", da ja die vorliegenden Programme, wie eingangs erwähnt, nur für das 20. Jahrhundert gelten.

Ein Beispiel für den immerwährenden Kalender (*Bild 1*):
Auf welchen Wochentag fällt der 1. April 1983?

```
0400  00      NOP
0401  21E404  LXI H 04E4
0404  DB02    IN ASHALT
0406  47      MOV B A
0407  DB01    IN BSHALT
0409  4F      MOV C A
040A  DB04    IN CSHALT
040C  E608    ANI 08
040E  C20A04  JNZ 040A
0411  DB04    IN CSHALT
0413  E608    ANI 08
0415  CA1104  JZ 0411
0418  DB02    IN ASHALT
041A  57      MOV D A
041B  79      MOV A C
041C  FE03    CPI 03
041E  D27404  JNC 0474
0421  C613    ADI 13
0423  4F      MOV C A
0424  7A      MOV A D
0425  C699    ADI 99
0427  27      DAA
0428  57      MOV D A
0429  97      SUB A
042A  D302    OUT LLAMPE
042C  1E05    MVI E 05
042E  23      INX H
042F  77      MOV M A
0430  1D      DCR E
0431  C22E04  JNZ 042E
0434  2B      DCX H
0435  2B      DCX H
0436  C5      PUSH B
0437  1E03    MVI E 03
0439  CD8804  CALL 0488
043C  1E41    MVI E 41
043E  CD8804  CALL 0488
0441  1E19    MVI E 19
0443  CD8804  CALL 0488
0446  C1      POP B
0447  51      MOV D C
0448  C5      PUSH B
0449  1E1E    MVI E 1E
044B  2B      DCX H
044C  97      SUB A
044D  77      MOV M A
044E  2B      DCX H
044F  CD8804  CALL 0488
0452  1E3C    MVI E 3C
0454  CD8804  CALL 0488
0457  C1      POP B
0458  50      MOV D B
0459  1E01    MVI E 01
045B  2B      DCX H
045C  2B      DCX H
045D  CD8804  CALL 0488
0460  2EE6    MVI L E6
0462  0600    MVI B 00
0464  0E93    MVI C 93
0466  CD7B04  CALL 047B
0469  23      INX H
046A  CD7B04  CALL 047B
046D  23      INX H
046E  CD7B04  CALL 047B
0471  D301    OUT RLAMPE
0473  76      HLT
0474  C601    ADI 01
0476  27      DAA
0477  4F      MOV C A
0478  C32904  JMP 0429
047B  7E      MOV A M
047C  81      ADD C
047D  27      DAA
047E  DA7C04  JC 047C
0481  05      DCR B
0482  F27C04  JP 047C
0485  91      SUB C
0486  47      MOV B A
0487  C9      RET
0488  7E      MOV A M
0489  82      ADD D
048A  27      DAA
048B  77      MOV M A
048C  0603    MVI B 03
048E  2B      DCX H
048F  7E      MOV A M
0490  CE00    ACI 00
0492  27      DAA
0493  77      MOV M A
0494  05      DCR B
0495  C28E04  JNZ 048E
0498  23      INX H
0499  23      INX H
049A  23      INX H
049B  1D      DCR E
049C  C28804  JNZ 0488
049F  23      INX H
04A0  C9      RET
```

Bild 1. Programm für einen (fast) immerwährenden Kalender

Eingabe: 1.4.1983 – Ausgabe: 0005 = Freitag.

Anleitung für die Bedienung des ITT-Mikrocomputers:

1. Programm laden
2. Reset drücken
3. Tageszahl mit A-Schalter eingeben
4. Monatszahl mit B-Schalter eingeben
5. Schalter „RUN" einmal takten
6. Jahreszahl mit A-Schalter eingeben
7. Schalter „RUN" einmal takten auf der rechten Anzeige erscheint die Codezahl W.

Der Biorhythmus spiegelt die wahrnehmbaren „Höhen" und „Tiefen" von Seele, Körper und Intellekt unserer Lebensenergie wider. Alle drei Zyklen beginnen mit dem Tage der Geburt und begleiten uns in sinusförmigem Verlauf ununterbrochen das ganze Leben hindurch. Aus diesem Grunde ist es möglich, auch unseren Biorhythmus im voraus zu berechnen [1]. Beispiel: geb. am 5.4.28, Biorhythmus für Juni 1979. Eingabe also: 5.4.28 – 1.6.79.

Für die Berechnung der Biorhythmen (*Bild 2*) wurde die gleiche gekürzte Formel wie zur Berechnung des Immerwährenden Kalenders herangezogen, nur daß jetzt N_1 für das Geburtsdatum, und N_2 für den 1. des zu untersuchenden Monats errechnet wird. Es erfolgt dann die Subtraktion $N_2 - N_1$ und man erhält die Differenz S der beiden Daten in Tagen. Dieses S wird anschließend jeweils durch 23, 28 und 33 dividiert. Die verbleibenden Reste sind dann die Anfangswerte für die entsprechenden Sinuskurven mit 23, 28 und 33 Tagen Periodendauer.

Anleitung für die Berechnung des Biorhythmus:

1. Programm laden
2. Reset drücken
3. Geburtstag mit A-Schalter eingeben
4. Geburtsmonat mit B-Schalter eingeben
5. Schalter „RUN" einmal takten
6. Geburtsjahr mit A-Schalter eingeben
7. Schalter „RUN" einmal takten
8. 1 mit A-Schalter eingeben
9. Zu untersuchender Monat mit B-Schalter eingeben
10. Schalter „RUN" einmal takten
11. Zu untersuchendes Jahr mit A-Schalter eingeben
12. Schalter „RUN" einmal takten, auf R-Lampen wird Anfangswert für den Körperzyklus angezeigt
13. Schalter „RUN" einmal takten, auf Anzeige erscheint Anfangswert für den Seelenzyklus
14. Schalter „RUN" einmal takten, es wird der Anfangswert für den Geistzyklus angezeigt.

Literatur

[1] *Hofer, R:* Biorhythmus- und Wochentags-Berechnung in BASIC. FUNKSCHAU 1978, Heft 22, Seite 1102.

```
0400  21EB04   LXI H 04EB
0403  CD3B04   CALL 043B
0406  CDA804   CALL 04A8
0409  2EE7     MVI L E7
040B  CD3B04   CALL 043B
040E  11EB04   LXI D 04EB
0411  2EEF     MVI L EF
0413  0603     MVI B 03
0415  37       STC
0416  3E99     MVI A 99
0418  CE00     ACI 00
041A  96       SUB M
041B  EB       XCHG
041C  86       ADD M
041D  27       DAA
041E  77       MOV M A
041F  EB       XCHG
0420  1B       DCX D
0421  2B       DCX H
0422  05       DCR B
0423  C21604   JNZ 0416
0426  EB       XCHG
0427  23       INX H
0428  112377   LXI D 7723
042B  CDD004   CALL 04D0
042E  112872   LXI D 7228
0431  CDD004   CALL 04D0
0434  113367   LXI D 6733
0437  CDD004   CALL 04D0
043A  76       HLT
043B  DB02     IN ASHALT
043D  47       MOV B A
043E  DB01     IN BSHALT
0440  4F       MOV C A
0441  CDA804   CALL 04A8
0444  DB02     IN ASHALT
0446  57       MOV D A
0447  79       MOV A C
0448  FE03     CPI 03
044A  D28D04   JNC 048D
044D  C613     ADI 13
044F  4F       MOV C A
0450  7A       MOV A D
0451  C699     ADI 99
0453  27       DAA
0454  57       MOV D A
0455  97       SUB A
0456  D302     OUT LLAMPE
0458  1E05     MVI E 05
045A  23       INX H
045B  77       MOV M A
045C  1D       DCR E
045D  C25A04   JNZ 045A
0460  2B       DCX H
0461  2B       DCX H
0462  C5       PUSH B
0463  1E03     MVI E 03
0465  CDB704   CALL 04B7
0468  1E41     MVI E 41
046A  CDB704   CALL 04B7
046D  1E19     MVI E 19
046F  CDB704   CALL 04B7
0472  C1       POP B
0473  51       MOV D C
0474  C5       PUSH B
0475  1E1E     MVI E 1E
0477  2B       DCX H
0478  97       SUB A
0479  77       MOV M A
047A  2B       DCX H
047B  CDB704   CALL 04B7
047E  1E3C     MVI E 3C
0480  CDB704   CALL 04B7
0483  C1       POP B
0484  50       MOV D B
0485  1E01     MVI E 01
0487  2B       DCX H
0488  2B       DCX H
0489  CDB704   CALL 04B7
048C  C9       RET
048D  C601     ADI 01
048F  27       DAA
0490  4F       MOV C A
0491  C35504   JMP 0455
0494  7E       MOV A M
0495  82       ADD D
0496  27       DAA
0497  DA9504   JC 0495
049A  4F       MOV C A
049B  78       MOV A B
049C  C699     ADI 99
049E  27       DAA
049F  47       MOV B A
04A0  79       MOV A C
04A1  F29504   JP 0495
04A4  83       ADD E
04A5  27       DAA
04A6  47       MOV B A
04A7  C9       RET
04A8  DB04     IN CSHALT
04AA  E608     ANI 08
04AC  C2A804   JNZ 04A8
04AF  DB04     IN CSHALT
04B1  E608     ANI 08
04B3  CAAF04   JZ 04AF
04B6  C9       RET
04B7  7E       MOV A M
04B8  82       ADD D
04B9  27       DAA
04BA  77       MOV M A
04BB  0603     MVI B 03
04BD  2B       DCX H
04BE  7E       MOV A M
04BF  CE00     ACI 00
04C1  27       DAA
04C2  77       MOV M A
04C3  05       DCR B
04C4  C2BD04   JNZ 04BD
04C7  23       INX H
04C8  23       INX H
04C9  23       INX H
04CA  1D       DCR E
04CB  C2B704   JNZ 04B7
04CE  23       INX H
04CF  C9       RET
04D0  0600     MVI B 00
04D2  CD9404   CALL 0494
04D5  23       INX H
04D6  CD9404   CALL 0494
04D9  23       INX H
04DA  CD9404   CALL 0494
04DD  D301     OUT RLAMPE
04DF  CDA804   CALL 04A8
04E2  2B       DCX H
04E3  2B       DCX H
04E4  C9       RET
```

Bild 2. Errechnung der drei Werte des Biorhythmus für ein bestimmtes Datum

Hans Werner

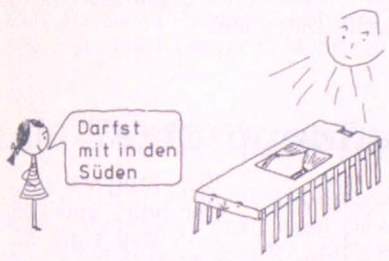
Mikrocomputer

Eine kleine Einführung

17. Teil

Löschen von EPROMs

Der Baustein ist mit einem Quarzglasfenster verschlossen und kann mit UV-Licht hoher Intensität gelöscht



werden. Durch Fotoströme wird das Gate gegen das Substrat (Trägermaterial) entladen. Normales Tageslicht reicht nicht aus.

RAM

Bei der Entwicklung eines Programms ist man noch nicht sicher, ob alle gewünschten Funktionen erfüllt sind. Zur Erprobung ist ein Speicher sinnvoll, dessen Daten beliebig geändert werden können.

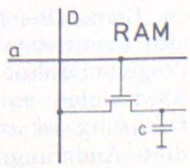
Nach der Erprobung kann entweder beim Hersteller ein ROM bestellt werden, oder der Anwender programmiert selbst ein PROM bzw. EPROM.

Weiterhin sind solche Typen als Datenspeicher im Mikrocomputersystem selbst notwendig.

Das RAM (Random Access Memory = Speicher mit wahlfreiem Zugriff) ist ein flüchtiger Speicher. Er verliert bei Stromausfall seine Daten. Dafür können sie jederzeit wieder eingelesen werden (Schreib-Lese-Speicher).

Dynamisches RAM

Jede Speicherzelle besteht aus einem Kondensator.



Schreiben: Mit der Adreßleitung a wird der Transistor in den niederohmigen Zustand geschaltet und über die Datenleitung die Information in den Kondensator geladen.
(1: C geladen; 0: C nicht geladen.)
Lesen: Bei niederohmigem Transistor steht die Information auf der Datenleitung.

Im Gegensatz zum EPROM ist die Ladung nicht auf dem vom Stromkreis sehr hochohmig isolierten Gate aufgebracht. Der Kondensator kann sich ständig über den Bahnwiderstand des Transistors entladen.

Um die Leckströme klein zu halten, sind für dynamische RAMs nur MOS-Techniken sinnvoll.

Etwa alle 2 ms muß die Ladung aufgefrischt werden (refreshing). Das geschieht z. B. durch periodisches Durchlaufen der Adreßleitungen. Das *Bild* zeigt nur eine Speicherzelle und nicht die zugehörige Auffrischschaltung.

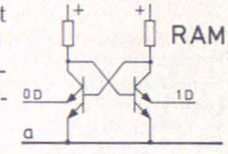
Dynamische Speicher müssen zyklisch aufgefrischt werden.

Statisches RAM

Statische Speicher benötigen keinen Auffrischzyklus.

Grundelement sind Flipflop-Schaltungen. Sie verlieren ihre Information nur bei Stromausfall.

In dem Beispiel ist eine Speicherzelle durch kreuzgekoppelte Multiemittertransistoren realisiert. Entweder ist der linke oder der rechte Transistor leitend.



Ruhezustand: a liegt an 0-Potential. Der Kollektorstrom des leitenden Transistors fließt über a ab.

0 schreiben: a an 1, Datenleitung 0D an 0 und 1D an 1. Der linke Transistor wird leitend.

1 schreiben: a an 1, Datenleitung 0D an 1 und 1D an 0. Der rechte Transistor wird leitend.

Lesen: 0D und 1D werden auf 0-Potential gezogen. Abhängig vom gespeicherten Zustand fließt über eine der beiden Datenleitungen ein Strom, der über eine Umschaltlogik das Potential 0 oder 1 der gemeinsamen Ausgangsleitung bestimmt.

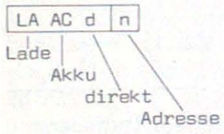
Nach der sicherlich nicht sehr interessanten Aufzählung von verschiedenen Speichertypen wollen wir unseren Mikrocomputer zum Leben erwecken. Dazu muß der grundsätzliche Ablauf der Befehlszyklen bekannt sein. Nicht verzweifeln, so genau wie beim ersten Modell ist das nicht nötig.

Der erste Befehl

soll die Transportanweisung sein:

> Lade den Inhalt der Speicherzelle n in den Akku.

Abkürzung:



Wir erinnern uns, die Abkürzungen müssen von einem Assemblerprogramm in die binäre Form des Maschinencode gebracht werden. Erst dieses Objektprogramm versteht der Mikrocomputer. Kleine Buchstaben sind in Wirklichkeit nicht erlaubt. Sie sollen uns nur die Gedächtnisarbeit erleichtern.

Beispiele:

LA	AC	d	00 09 H

Mit dem Befehl gelangt der Inhalt der Zelle 00 09 H in den Akkumulator. Das H deutet auf eine hexadezimale Zahl hin. 00 09 H entspricht der Binärzahl 00000000 00001001 und der Dezimalzahl 9.

00 0A H entspricht dem dezimalen Wert 10.

Der Ladebefehl besteht aus 3 Byte:
a) Operationsteil: LA AC d 8 bit Speicherzelle 2
b) hoher Adreßteil: 00 8 bit Speicherzelle 3
c) tiefer Adreßteil: 09 8 bit Speicherzelle 4

Er kann nicht in einem Befehlszyklus ausgeführt werden, sondern er erfordert 4 vollständige Zyklen.

Zyklus 1
Der Programmzähler PZ erreicht Adresse 2, und der Inhalt der angesprochenen Speicherzelle gelangt in das Befehlsregister.
Am Schluß springt PZ auf 3.

Zyklus 2
Der hohe Teil der Adresse wird vom Datenregister übernommen.

Danach wird PZ wieder um 1 erhöht.

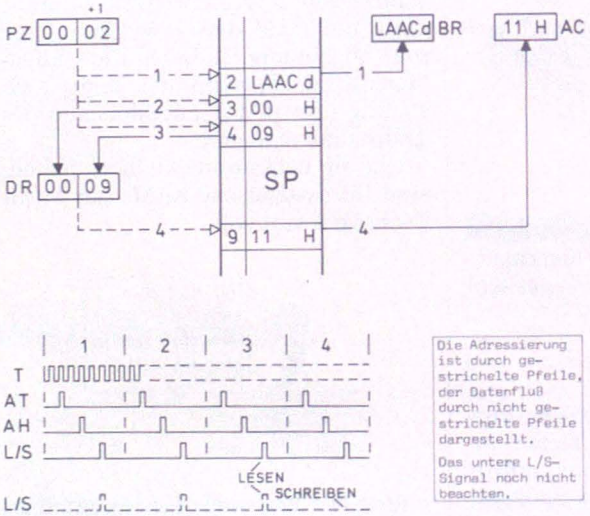
Zyklus 3
Der tiefe Teil der ...

Zyklus 4
Das Datenregister adressiert Speicherzelle 9, deren Inhalt in den Akkumulator geladen wird.

In jedem Zyklus wird mit Adresse AT der Speicherplatz und mit AH das ROM angesprochen (A9, A8 = 0).

Die Flußrichtung der Daten legt Impuls L/S fest.

L/S = 1 bedeutet: Lesen bzw. die CPU nimmt Daten auf.



> Ein Stolperstein:
> Der hohe und tiefe Adreßteil eines Befehls kann leicht mit den Adreßhälften AH, AT verwechselt werden.
> Mit AH und AT wird der Speicher adressiert (siehe Impulsschema).

(*Fortsetzung folgt*)

Berichtigung

Programm zur Spulenberechnung mit dem TI-59

FUNKSCHAU 1979,
Heft 7, Seite 400

Wie uns der Autor, Fred H. Heller, mitteilt, haben sich in die Formeln drei Fehler eingeschlichen. Die Einheit nH muß in der Formel für L selbstverständlich im Zähler stehen. Die Einheit cm darf in den Formeln für l_{sp} und D natürlich nicht mehr unter der Wurzel stehen. Wir bitten um Ihr Verständnis, daß uns die Wurzel-Striche ein paar mm zu lang geraten sind.

KIM- und SYM-Monitor-Unterprogramme

Anwenderprogramme lassen sich meist wesentlich vereinfachen, wenn man Unterprogramme verwendet, die sich im ROM des Mikrocomputers als fest gespeicherte Monitor-Routinen befinden. Für die verbreiteten Systeme KIM-1 und SYM-1 [1] haben wir die interessantesten zusammengestellt (*Tabelle*). KIM und SYM sind zwar weitgehend software-kompatibel, aber eben nur weitgehend [2, 3].

Zum einen werden bei SYM-Unterprogrammen meist mehr Register „gerettet" als beim KIM; zum anderen befinden sich aber bestimmte Adressen, z. B. die Display-Bufferzellen, beim SYM nicht im Zero-Page-Bereich, sondern oberhalb A000. Das führt dazu, daß es recht zeitraubend sein kann, KIM-Programme für den SYM umzuschreiben, da aus Zwei-Byte-Befehlen u. U. Drei-Byte-Befehle werden, was die Änderung der meisten Sprungadressen bzw. der Branch-Befehle erfordert. Es wäre also übertrieben zu sagen, daß KIM-Programme ohne weiteres auch auf dem SYM laufen; ein wenig Programmier-Erfahrung ist schon notwendig, um diese Änderungen vorzunehmen.

Da die Register bei vielen SYM-Unterprogrammen auf den Stack („Kellerspeicher") gerettet werden, benötigt der SYM auch mehr Speicherzellen im Stackbereich, der sich im Bereich 0100...01FF befindet. KIM-Programme, die sich im Stack-Adressenbereich befinden, werden also u. U. von SYM-Monitor-Unterprogrammen zum Teil überschrieben, wenn sie recht weit in diesen Adressenraum hineinreichen.

Es soll aber nicht unerwähnt bleiben, daß die SYM-Unterprogramme trotz allem deutlich komfortabler ausgelegt sind. Eine Reihe anderer nützlicher SYM-Routinen, die hier nicht verzeichnet sind, existieren nicht im KIM-Monitor. Fe.

KIM-1			SYM-1			Monitor-Unterprogramme
Name	Adresse	Verlorene Register	Name	Adresse	Verlorene Register	Wirkung
SCAND	1F19	A, X, Y	OUTDSP	89C1	–	Anzeige einer Adresse mit Dateninhalt
SCANDS	1F1F	A, X, Y	SCAND	8906	A	Anzeige von drei Bytes im Display
INCPNT	1F63	–	INCCMP	82B2	–	Erhöhen der Adresse um 1
GETKEY	1F6A	A, X, Y	GETKEY	88AF	–	Tastaturabfrage
GETBYT	1F9D	A, Y	INBYTE	81D9	A	Ein Byte vom Terminal holen
PRTBYT	1E3B	A, Y	OUTBYT	82FA	–	Ein Byte drucken
OUTCH	1EAO	A, Y	OUTCHR	8A47	–	Ein ASCII-Zeichen drucken
GETCH	1E5A	A, Y	INTCHR	8A58	–	Ein ASCII-Zeichen holen
CRLF	1E2F	A, X, Y	CRLF	834D	–	Neue Zeile beginnen
PRT PNT	1E1E	A, Y	OUTS 2	8319	A, X	Adresse ausdrucken

Literatur

[1] Monitor-Befehle des Mikrocomputers SYM-1. FUNKSCHAU 1979, Heft 1, S. 18
[2] KIM-1-Handbuch (Astronic, München)
[3] SYM-1-Handbuch (Astronic, München)

RAM-Testprogramm für den Z-80-KIT

Oft erweitert man den RAM-Bereich seines Mikroprozessors, weiß aber nicht, ob die zugekauften RAM-ICs in Ordnung sind. Die neuen Speicherplätze einzeln zu überprüfen, wäre zu mühsam und zeitaufwendig. Das hier beschriebene Programm führt diese Überprüfung automatisch und in kurzer Zeit durch (6 s für 1 KByte). Das *Bild* zeigt das Flußdiagramm.

Die Anfangsadresse des zu überprüfenden Speicherbereiches muß in die Register H und L, die Endadresse in die Register D und E eingegeben werden, wobei beachtet werden sollte, daß stets DE ≧ HL ist; sonst läuft das Programm über die Adresse 0000 und löscht sich bei Erreichen seiner eigenen Adresse selbst.

Nach dem Start lädt das Programm den durch HL adressierten Speicherplatz mit den Daten 00 bis FF und überprüft nach jedem Laden, ob der Speicherplatz auch diese Daten enthält. Wird diese Überprüfung immer bejaht, so wird HL um 1 erhöht, und das Programm überprüft nun diesen nächsten Speicherplatz. Das geschieht so lange, bis der Wert von HL den Wert von DE erreicht hat. Danach springt das Programm in die Anzeigeroutine des Z-80-KIT; es ist mit der Anzeige der Endadresse beendet.

Wird die Überprüfung des Speicherplatzes verneint, d. h. enthält der Speicherplatz nicht die Daten, mit denen er vorher geladen worden ist, so springt das Programm ebenfalls in die Anzeigeroutine, und die Adresse des „defekten" Speicherplatzes wird im Adressenanzeigeteil angezeigt.

Jörg Ullrich

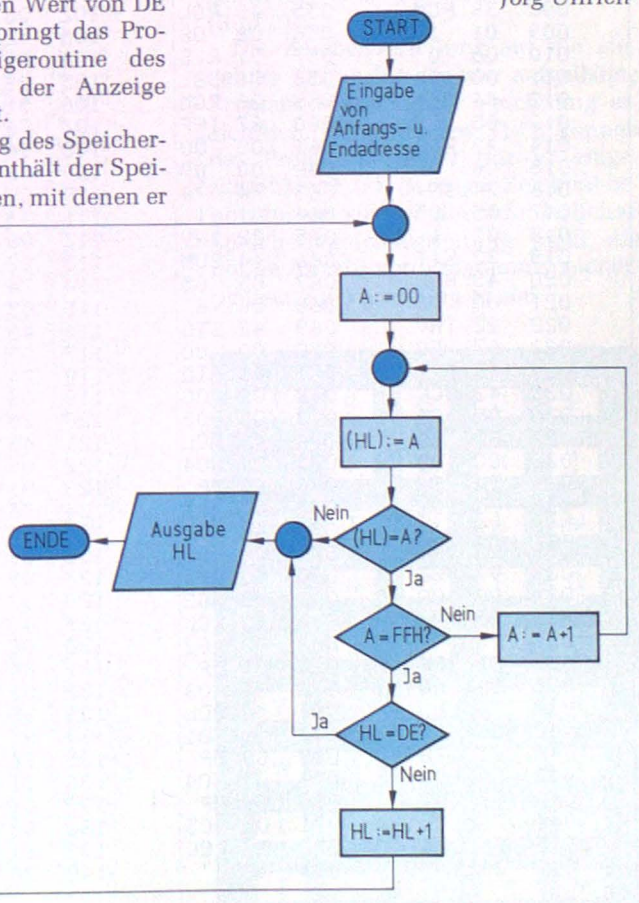
Anmerkung der Redaktion:

Über RAM-Testmethoden gibt es die unterschiedlichsten Auffassungen und Ideologien. Fest steht, daß 95 % aller Fehler schon dadurch gefunden werden können, indem jede Speicherzelle wechselweise mit 00 und FF beaufschlagt wird. Die hier vorgestellte Methode findet bestimmt nicht alle übrigen möglichen Fehler, stellt aber wegen der geringen erforderlichen Prüfzeit eine universell verwendbare Version dar.

Maschinencode für den Z-80-KIT

Stack-Pointer 3CFFH
Startadresse 3C50H

Adresse	Code
3C50	3E00
3C52	77
3C53	BE
3C54	C20D01
3C57	FEFF
3C59	2803
3C5B	3C
3C5C	18F4
3C5E	7C
3C5F	BA
3C60	2005
3C62	7D
3C63	BB
3C64	CA0D01
3C67	23
3C68	18E6



Flußdiagramm des RAM-Tests. Das Registerpaar HL enthält die Anfangsadresse des zu prüfenden Speicherbereichs. DE beinhaltet die Endadresse. Mit A ist der Akkumulator und mit (HL) der durch HL adressierte Speicherplatz bezeichnet

Joachim Schmitt

Taschenrechner als Schreibmaschine

Das vorliegende Programm vereinfacht die Druckmöglichkeiten des programmierbaren Taschenrechners TI-59. Dabei werden die Buchstaben, Zahlen und Zeichen wie bei einer Schreibmaschine über die Tastatur eingegeben.

Nach dem Einprogrammieren *(Bild 1)* (eventuell Karte einlesen) wird zunächst „Start/Reset" (2nd E') betätigt. Dies ist nur einmal, nämlich direkt nach dem Einlesen erforderlich. Vor dem „Schreiben" einer jeden Stelle muß jeweils die Taste SBR (= Unterprogramm) betätigt werden. Bei den Buchstaben A bis E ist dies nicht nötig.

Um die Darstellung mehrerer zusammenhängender Leerzeichen zu erleichtern, muß folgendes beachtet werden: Die Anzahl der gewünschten Leerzeichen eingeben, dann SBR +/– betätigen. Nach jedem Schritt wird angezeigt, wie viele Zeichen noch in der Zeile dargestellt werden können.

Muß ein Schritt wegen Falschdruck redigiert werden, so wird „Korrektur" (2nd A') gedrückt und das richtige Zeichen eingegeben. Die Korrektur kann

```
000  22 INV     047  97 DSZ     094  47 CMS     141  12  B     188  06  6     235  76 LBL
001  77  GE     048  09  09     095  06  6      142  01  1     189  81 RST     236  55  ÷
002  96 WRT     049  00  00     096  42 STO     143  04  4     190  76 LBL     237  04  4
003  42 STO     050  55  55     097  00  00     144  81 RST     191  34 fX      238  01  1
004  06  06     051  00  0      098  04  4      145  76 LBL     192  02  2     239  81 RST
005  22 INV     052  61 GTO     099  42 STO     146  13  C     193  07  7     240  76 LBL
006  97 DSZ     053  00  00     100  05  05     147  01  1     194  81 RST     241  61 GTO
007  00  00     054  03  03     101  02  2      148  05  5     195  76 LBL     242  04  4
008  36 PGM     055  43 RCL     102  00  0      149  81 RST     196  35 1/X     243  02  2
009  01  1      056  08  08     103  42 STO     150  76 LBL     197  03  3     244  81 RST
010  00  0      057  91 R/S     104  08  08     151  14  D     198  00  0     245  76 LBL
011  00  0      058  76 LBL     105  29 CP      152  01  1     199  81 RST     246  71 SBR
012  64 PD*     059  36 PGM     106  91 R/S     153  06  6     200  76 LBL     247  04  4
013  05  05     060  87 IFF     107  76 LBL     154  81 RST     201  42 STO     248  03  3
014  43 RCL     061  00  00     108  96 WRT     155  76 LBL     202  03  3     249  81 RST
015  06  06     062  00  00     109  42 STO     156  15  E     203  01  1     250  76 LBL
016  74 SM*     063  74  74     110  07  07     157  01  1     204  81 RST     251  81 RST
017  05  05     064  01  1      111  43 RCL     158  07  7     205  76 LBL     252  04  4
018  01  1      065  22 INV     112  06  06     159  81 RST     206  43 RCL     253  04  4
019  32 X:T     066  44 SUM     113  22 INV     160  76 LBL     207  03  3     254  81 RST
020  43 RCL     067  05  05     114  74 SM*     161  22 INV     208  02  2     255  76 LBL
021  00  00     068  06  6      115  05  05     162  02  2     209  81 RST     256  86 STF
022  22 INV     069  42 STO     116  43 RCL     163  01  1     210  76 LBL     257  04  4
023  67  EQ     070  00  00     117  07  07     164  81 RST     211  44 SUM     258  05  5
024  19 D'      071  61 GTO     118  74 SM*     165  76 LBL     212  03  3     259  81 RST
025  43 RCL     072  00  00     119  05  05     166  23 LNX     213  03  3     260  76 LBL
026  05  05     073  05  05     120  29 CP      167  02  2     214  81 RST     261  91 R/S
027  67  EQ     074  43 RCL     121  43 RCL     168  02  2     215  76 LBL     262  04  4
028  00  00     075  04  04     122  08  08     169  81 RST     216  45 Y^      263  06  6
029  74  74     076  69 OP      123  91 R/S     170  76 LBL     217  03  3     264  81 RST
030  76 LBL     077  01  01     124  76 LBL     171  24 CE      218  04  4     265  76 LBL
031  19 D'      078  43 RCL     125  16 A'      172  02  2     219  81 RST     266  87 IFF
032  22 INV     079  03  03     126  09  9      173  03  3     220  76 LBL     267  02  2
033  97 DSZ     080  69 OP      127  09  9      174  81 RST     221  52 EE      268  81 RST
034  08  08     081  02  02     128  32 X:T     175  76 LBL     222  03  3     269  76 LBL
035  95  =      082  43 RCL     129  43 RCL     176  25 CLR     223  05  5     270  88 DMS
036  22 INV     083  02  02     130  08  08     177  02  2     224  81 RST     271  03  3
037  87 IFF     084  69 OP      131  85  +      178  04  4     225  76 LBL     272  81 RST
038  00  00     085  03  03     132  01  1      179  81 RST     226  53  (     273  76 LBL
039  76 LBL     086  43 RCL     133  95  =      180  76 LBL     227  03  3     274  89  π
040  00  0      087  01  01     134  91 R/S     181  32 X:T     228  06  6     275  04  4
041  61 GTO     088  69 OP      135  76 LBL     182  02  2     229  81 RST     276  81 RST
042  00  00     089  04  04     136  11  A      183  05  5     230  76 LBL     277  76 LBL
043  03  03     090  69 OP      137  01  1      184  81 RST     231  54  )     278  77  GE
044  76 LBL     091  05  05     138  03  3      185  76 LBL     232  03  3     279  05  5
045  76 LBL     092  76 LBL     139  81 RST     186  33 X²      233  07  7     280  81 RST
046  22 INV     093  10 E'      140  76 LBL     187  02  2     234  81 RST     281  76 LBL
```

Bild 1. Zugegeben – kurz ist das Programm gerade nicht, aber wenn es erst einmal auf einer Magnetkarte ist, erleichtert es den Ausdruck von Kommentaren zu Programmen auf dem PC-100 A/B ganz erheblich

jedoch nur für jeweils eine Stelle erfolgen.

Ist die Zeile vollständig, so erfolgt automatisch der Ausdruck. Soll der Ausdruck erfolgen, bevor die Zeile mit zwanzig Zeichen gefüllt ist, muß „Drucken" (2nd C') betätigt werden. Dies geht schneller, als den Rest der Zeile mit Leerzeichen zu füllen.

Soll die zuletzt ausgedruckte Zeile wiederholt werden, so wird „letzte Zeile drucken" (2nd B') betätigt. Will man eine unvollständige, noch nicht ausgedruckte Zeile löschen, also von neuem beginnen, so ist „Start/Reset" (2nd E') zu drücken. Das Programm setzt daraufhin auf Ausgangsbedingungen zurück. Dennoch kann die Funktion „letzte Zeile drucken" weiterhin ausgeführt werden.

Kurzerläuterung zum Programm „Schreibmaschine"

Schritt	Beschreibung	Eingabe	Befehl	Anzeige
1	Eingeben des Programms bzw. Karte einlesen			
2	Start/Reset betätigen		2nd E'	20
3 a	Symbol eingeben (SBR nicht nötig bei A...E)		SBR (2nd) ...	
b	n Leerzeichen eingeben	n	SBR +/–	
4 a	Korrektur einer falschen Eingabe		2nd A'	
b	Korrigierte Eingabe siehe Schritt 3			
5	Vorzeitiger Ausdruck		2nd C'	20
6	Wiederholung der letzten Druckzeile		2nd B'	
7	Ganze Zeile löschen (Die Schritte 3 bis 7 können in beliebiger Reihenfolge wiederholt werden.)		2nd E'	20

```
282  78  Σ+
283  06   6
284  81  RST
285  76  LBL
286  79   x̄
287  07   7
288  81  RST
289  76  LBL
290  67   EQ
291  08   8
292  81  RST
293  76  LBL
294  68  NOP
295  09   9
296  81  RST
297  76  LBL
298  69   DP
299  01   1
300  02   2
301  81  RST
302  76  LBL
303  97  DSZ
304  01   1
305  81  RST
306  76  LBL
307  27  INV
308  07   7
309  04   4
310  81  RST
311  76  LBL
312  28  LOG
313  05   5
314  04   4
315  81  RST
316  76  LBL
317  29   CP
318  06   6
319  05   5
320  81  RST
321  76  LBL
322  20  CLR
323  06   6
324  01   1
325  81  RST
326  76  LBL
327  37  P/R
328  06   6
329  02   2
330  81  RST
331  76  LBL
332  38  SIN
333  07   7
334  05   5
335  81  RST
336  76  LBL
337  39  COS
338  05   5
339  02   2
340  81  RST
341  76  LBL
342  30  TAN
343  05   5
344  03   3
345  81  RST
346  76  LBL
347  47  CMS
348  06   6
349  06   6
350  81  RST
351  76  LBL
352  48  EXC
353  07   7
354  00   0
355  81  RST
356  76  LBL
357  49  PRD
358  06   6
359  07   7
360  22  INV
361  76  LBL
362  99  PRT
363  06   6
364  00   0
365  81  RST
366  76  LBL
367  57  ENG
368  07   7
369  07   7
370  81  RST
371  76  LBL
372  58  FIX
373  05   5
374  05   5
375  81  RST
376  76  LBL
377  59  INT
378  05   5
379  06   6
380  81  RST
381  76  LBL
382  50  IxI
383  07   7
384  02   2
385  81  RST
386  76  LBL
387  66  PAU
388  07   7
389  06   6
390  81  RST
391  76  LBL
392  65   ×
393  05   5
394  00   0
395  81  RST
396  76  LBL
397  60  DEG
398  05   5
399  01   1
400  81  RST
401  76  LBL
402  75   -
403  02   2
404  00   0
405  81  RST
406  76  LBL
407  70  RAD
408  06   6
409  03   3
410  81  RST
411  76  LBL
412  85   +
413  04   4
414  07   7
415  81  RST
416  76  LBL
417  80  GRD
418  07   7
419  03   3
420  81  RST
421  76  LBL
422  93   .
423  04   4
424  00   0
425  81  RST
426  76  LBL
427  98  ADV
428  05   5
429  07   7
430  81  RST
431  76  LBL
432  95   =
433  06   6
434  04   4
435  81  RST
436  76  LBL
437  90  LST
438  07   7
439  01   1
440  81  RST
441  76  LBL
442  94  +/-
443  85   +
444  01   1
445  95   =
446  42  STO
447  09   09
448  22  INV
449  86  STF
450  00   00
451  61  GTO
452  76  LBL
453  76  LBL
454  17  B'
455  69  DP
456  05   05
457  91  R/S
458  76  LBL
459  18  C'
460  86  STF
461  00   00
462  00   0
463  61  GTO
464  00   00
465  03   03
466  00   0
467  00   0
468  00   0
469  00   0
470  00   0
```

Zur Erleichterung des Auffindens der entsprechenden Tasten kann die jedem TI-58/59 beiliegende Tastencode-Schablone insofern abgeändert werden, daß man über den Tastenausschnitten kleine Klebefolien aufbringt und darauf die zu druckenden Symbole zeichnet, beispielsweise die Erstfunktion schwarz, die Zweitfunktion blau *(Bild 2)*.

Die Zeichenverteilung auf die einzelnen Tasten für das hier aufgelistete Programm ist aus der Zeichnung ersichtlich. Benutzer des TI-58 können das Programm leider nur in eingeschränkter Form (weniger Zeichen) benutzen, da sie bei der erforderlichen Speicherbereichsverteilung nicht die nötige Anzahl von Programmspeicherplätzen zur Verfügung haben.

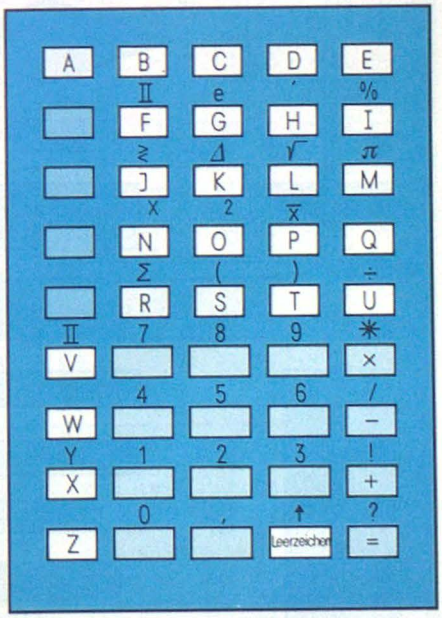


Bild 2. Tastencode-Schablone für den TI-59

Kalender-Algorithmus

In dem Beitrag in Heft 22/1978, S. 1102, ist bei der Angabe der Schaltregel für den Gregorianischen Kalender ein Fehler unterlaufen. Daß alle durch 4 teilbaren Jahre Schaltjahre sind, ist allgemein bekannt. Weniger geläufig ist jedoch die durch die Gregorianische Kalenderreform 1582 eingeführte Ausnahme, daß von den Jahrhundertjahren nur diejenigen Schaltjahre sein sollen, die durch 400 teilbar sind. 1600, 2000, 2400 sind also Schaltjahre, 1700, 1800, 1900, 2100, 2200 sind keine Schaltjahre.

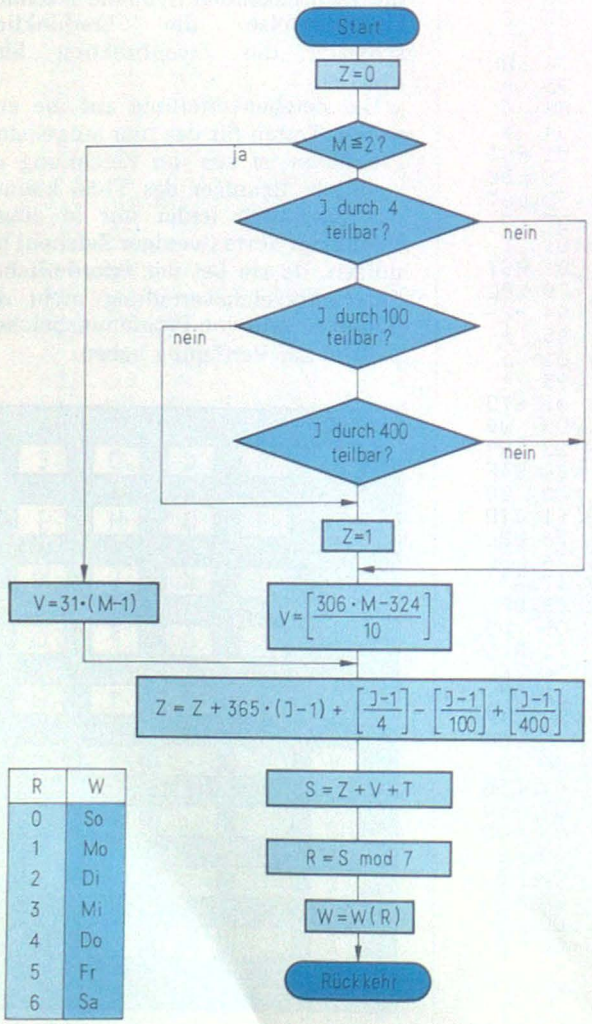
Ziel der Reform war es, das aus ganzen Tagen bestehende Kalenderjahr dem nicht in ganzen Tagen meßbaren Lauf der Erde um die Sonne besser anzupassen, so daß z. B. der Frühlingsanfang am 21. März erhalten bleibt.

Entsprechend dem Fehler in der Schaltregel ist in Heft 22 auch der Gültigkeitsbereich für das Programm falsch angegeben: Nicht vom 1. 3. 1600 bis 28. 2. 2000, sondern vom 1. 3. 1900 bis 28. 2. 2100 ist das Programm anwendbar.

Die Korrekturen seien zum Anlaß genommen, einen Kalender-Algorithmus vorzustellen, der keinen eingeschränkten Gültigkeitsbereich hat (schon in Heft 22 wurde erwähnt, daß dafür ein nur geringer Mehraufwand nötig ist) und der für die Ermittlung der bis zum Vormonat des Monats M vergangenen Tage V keine Tabelle, sondern eine für Mikrocomputer geeignete Formel benutzt (*Bild 1*). Im folgenden soll nicht das gesamte Biorhythmus-Programm wiederholt werden, sondern nur eine Alternative zu dem separat verwendbaren Kalender-Unterprogramm angegeben werden (*Bild 2*). Damit das Programm auch ohne Kenntnis von Heft 22 verständlich wird, seien zuvor noch einige Abkürzungen und Symbole erläutert. Zu den gegebenen Werten Tag T, Monat M, Jahr J (vierstellig) eines Datums im Gregorianischen Kalender werden die fortlaufende Tagesnummer S und der Wochentag W berechnet. Die fortlaufende Tagesnummer ist gleichzeitig die Anzahl der von einem Bezugspunkt Null bis zum gegebenen Datum vergangenen Tage. Sie ist ein Hilfsmittel für die Ermittlung der exakten Differenz (in Tagen) zwischen zwei gegebenen Daten (z. B. Geburtstag und heutiger Tag).

Zum Flußdiagramm sei noch vermerkt, daß die eckigen Klammern angeben, daß nur der ganze Teil des Ergebnisses zu nehmen ist, Stellen nach dem Komma bzw. Bruchteile also wegzulassen sind. Und „mod 7" gibt an, daß nur der bei Division durch 7 sich ergebende Rest zu nehmen ist. Das BASIC-Programm ist an die Form von Heft 22 angelehnt, damit interessierten Lesern der Vergleich erleichtert wird.

Jindra Singer



◄ Bild 1. Flußdiagramm des Kalender-Algorithmus

```
Ø11Ø INPUT T,M,J
Ø12Ø Z=Ø
Ø122 IF M<=2 THEN 296
Ø13Ø IF J/4<>INT(J/4) THEN 292
Ø132 IF J/1ØØ<>INT(J/1ØØ) THEN 29Ø
Ø134 IF J/4ØØ<>INT(J/4ØØ) THEN 292
Ø29Ø Z=1
Ø292 V=INT((3Ø6*M-324)/1Ø)
Ø294 GO TO 3ØØ
Ø296 V=(M-1)*31
Ø3ØØ Z=Z+(J-1)*365+INT((J-1)/4)
Ø3Ø1 Z=Z-INT((J-1)/1ØØ)+INT((J-1)/4ØØ)
Ø34Ø S=Z+V+T
Ø35Ø R=S-INT(S/7)*7
Ø37Ø FOR I=1TO7
Ø38Ø READ W$(I)
Ø39Ø NEXT I
Ø4ØØ PRINT W$(R+1)
Ø51Ø DATA SONNTAG,MONTAG,DIENSTAG,MITTWOCH,
     DONNERSTAG,FREITAG,SAMSTAG
```

Bild 2. Dieses BASIC-Programm berücksichtigt auch die „Ausnahme von der Regel" im Gregorianischen Kalender

Interrupt-Uhr mit dem KIM-1

Mikrocomputer werden oft zur automatischen Meßwertverarbeitung herangezogen, und außer der Messung analoger Eingangsgrößen spielt die Zeit dabei eine wichtige Rolle. Versteht man unter „Zeit" die tatsächliche Uhrzeit, so spricht man auch von „Echtzeit". Sinnvollerweise beauftragt man den Mikrocomputer selbst damit, die aktuelle Uhrzeit bereitzustellen.

Zur Realisation einer reinen Software-Uhr gibt es mehrere Möglichkeiten. Einmal könnte man eine Verzögerungsschleife so programmieren, daß sie einige Speicherzellen für Stunden, Minuten usw. fortlaufend weiterstellt, so daß diese die Uhrzeit enthalten. Der Nachteil eines solchen Verfahrens ist so groß, daß es in der Praxis kaum angewendet wird: Der Mikrocomputer ist vollständig damit beschäftigt, die Uhr weiterlaufen zu lassen, und hat z. B. für die Verarbeitung parallel laufender Meßvorgänge keine Zeit mehr.

Die Alternative ist die Ausnutzung des Prozessor-Interrupts. Es ist z. B. möglich, jede volle Sekunde einen Interrupt auszulösen; der Prozessor springt aus dem parallel laufenden Hauptprogramm in eine Interrupt-Routine, die die Uhr um eine Sekunde weiterstellt, und kehrt dann wieder in das Hauptprogramm zurück. Da der relative Zeitbedarf für die Interrupt-Routine minimal ist, kann das Hauptprogramm ungestört z. B. Meßwerte verarbeiten oder mit einem Terminal kommunizieren; es kann aber auch die Uhrzeit auf einem Siebensegment-Display darstellen. Genau dies tut das im *Bild* aufgelistete 6502-Programm.

Das (KIM-spezifische) Hauptprogramm (03A0...03C0) dient lediglich dazu, den Inhalt der Zellen 00DC, 00DD und 00DE auf dem sechsstelligen KIM-Display als Stunden, Minuten und Sekunden darzustellen. Der Tag-Zähler (00DB) wird hier nicht angezeigt, ebenso nicht der Viertelsekunden-Zähler (00DF).

Warum eigentlich ein Viertelsekunden-Zähler? Nun, der Mikrocomputer KIM-1 besitzt einen wunderschönen 8-bit-Timer, mit dem man nach einer vorprogrammierten Zeit einen Interrupt auslösen kann (vgl. FUNKSCHAU 1979, Heft 8, Seite 466, „Gebrauchsanleitung für den KIM-Timer"). Er läßt sich maximal aber nur mit einer Zeit von 255 x 1024 µs voreinstellen, so daß der Interrupt nicht jede Sekunde, sondern bereits nach einer Viertelsekunde stattfinden muß. Deswegen fungiert eine Zelle im Zero-Page-Bereich (00DF) als Viertelsekunden-Zähler; ihr Inhalt wird normalerweise nicht weiterverwendet.

Ein ähnliches Programm („Clock") wurde bereits im „First Book of KIM" von Charles Parsons veröffentlicht; die Interrupt-Routine umfaßt dabei hexadezimal 5B Bytes. Wendet man dagegen die hervorragenden Möglichkeiten der indizierten Adressierung an, so läßt sich diese Routine auf (hex) 28 Bytes

```
03A0 A9 03 8D FB 17 A9 D8 8D
03A8 FA 17 8D 0E 17 A5 DC 85
03B0 FB A5 DD 85 FA A5 DE 85
03B8 F9 20 1F 1F 20 6A 1F C9
03C0 00 D0 EA 4C 25 19

03D3 FF TAGE
03D4 24 STUNDEN
03D5 60 MINUTEN
03D6 60 SEKUNDEN
03D7 04 1/4 SEK.

03D8 48       PHA
03D9 8A       TXA
03DA 48       PHA
03DB A2 10    LDX #10
03DD CA       DEX
03DE D0 FD    BNE 03DD
03E0 A9 F4    LDA #F4
03E2 8D 0F 17 STA 170F
03E5 F8       SED
03E6 A2 05    LDX #05
03E8 CA       DEX
03E9 30 10    BMI 03FB
03EB B5 DB    LDA DB,X
03ED 18       CLC
03EE 69 01    ADC #01
03F0 DD D3 03 CMP 03D3,X
03F3 90 02    BCC 03F7
03F5 A9 00    LDA #00
03F7 95 DB    STA DB,X
03F9 F0 ED    BEQ 03E8
03FB D8       CLD
03FC 68       PLA
03FD AA       TAX
03FE 68       PLA
03FF 40       RTI

00DB 04 TAG
00DC 19 STUNDE
00DD 01 MINUTE
00DE 35 SEKUNDE
00DF 02 1/4 SEK.
```

Programm für die Interrupt-Uhr. Das Hauptprogramm springt beim Drücken der KIM-Taste 0 zum Monitor-Programm und erlaubt das Programmieren, während die Uhr weiterläuft. Die „Kaltstartadresse" ist 03A0; bei laufender Uhr kann man vom Monitorprogramm aus über die Adresse 03AD wieder in das Display-Programm zurück

FUNKSCHAU-Mikro-Poster

6502-Operationscodes

MN.	/IM	/AB	/ZP	/AC	/X)	/)Y	/ZX	/ZY	/AX	/AY
ADC	69	6D	65	--	61	71	75	--	7D	79
AND	29	2D	25	--	21	31	35	--	3D	39
ASL	--	0E	06	0A	--	--	16	--	1E	--
BIT	--	2C	24	--	--	--	--	--	--	--
CMP	C9	CD	C5	--	C1	D1	D5	--	DD	D9
CPX	E0	EC	E4	--	--	--	--	--	--	--
CPY	C0	CC	C4	--	--	--	--	--	--	--
DEC	--	CE	C6	--	--	--	D6	--	DE	--
EOR	49	4D	45	--	41	51	55	--	5D	59
INC	--	EE	E6	--	--	--	F6	--	FE	--
LDA	A9	AD	A5	--	A1	B1	B5	--	BD	B9
LDX	A2	AE	A6	--	--	--	--	B6	--	BE
LDY	A0	AC	A4	--	--	--	B4	--	BC	--
LSR	--	4E	46	4A	--	--	56	--	5E	--
ORA	09	0D	05	--	01	11	15	--	1D	19
ROL	--	2E	26	2A	--	--	36	--	3E	--
ROR	--	6E	66	6A	--	--	76	--	7E	--
SBC	E9	ED	E5	--	E1	F1	F5	--	FD	F9
STA	--	8D	85	--	81	91	95	--	9D	99
STX	--	8E	86	--	--	--	--	96	--	--
STY	--	8C	84	--	--	--	94	--	--	--

BCC	90	BCS	B0	BEQ	F0	BMI	30			
BNE	D0	BPL	10	BRK	00	BVC	50			
BVS	70	CLC	18	CLD	D8	CLI	58			
CLV	B8	DEX	CA	DEY	88	INX	E8			
INY	C8	JMI	6C	JMP	4C	JSR	20			
NOP	EA	PHA	48	PHP	08	PLA	68			
PLP	28	RTI	40	RTS	60	SEC	38			
SED	F8	SEI	78	TAX	AA	TAY	A8			
TSX	BA	TXA	8A	TXS	9A	TYA	98			

Adressierungsarten: IM = Immediate, AB = Absolut, ZP = Zero Page, AC = Accu, X) = Indirekt X,)Y = Indirekt Y, ZX = Zero Page X, ZY = Zero Page Y, AX = Absolut X, AY = Absolut Y

verkürzen, also um mehr als die Hälfte. Das Weiterzählen der aktuellen Zeit geschieht nicht für jede Zelle (00DB...00DF) einzeln, sondern in einer Schleife, bei der das X-Register dazu dient, Zero-Page-Adressen zu inkrementieren und mit den im Bereich 03D3...03D7 abgelegten Grenzwerten zu vergleichen. So werden etwa die Tage nur dann weitergezählt, wenn bei den Stunden der Grenzwert 24 erreicht ist; der Stundenzähler wird dann gleichzeitig auf Null rückgesetzt.

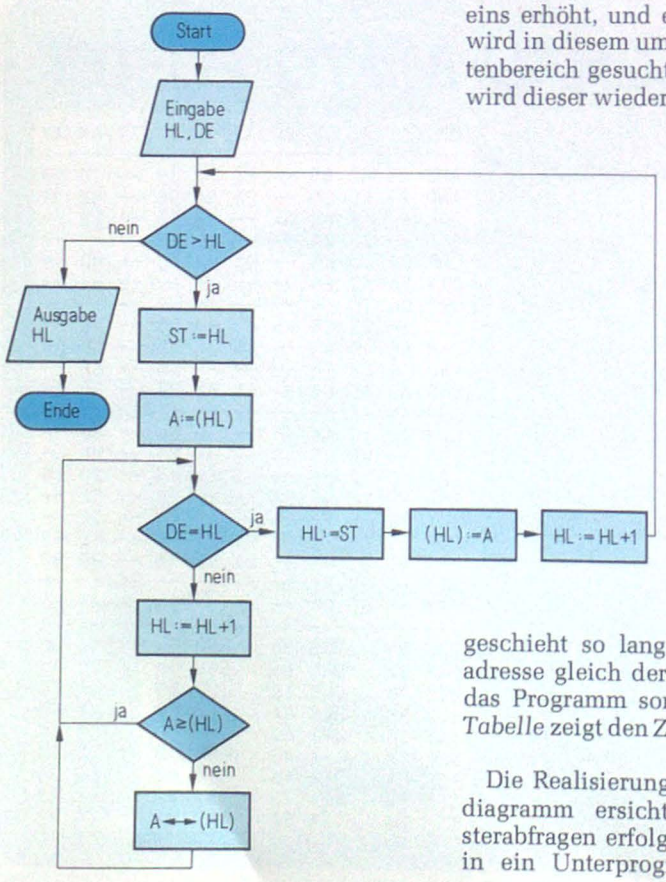
Der Grenzwert für die Tage ist im Beispielprogramm FF, was dazu führt, daß – weil diese Zahl im Dezimal-Modus des 6502 nie erreicht wird – bis 99 Tage gezählt werden kann. Selbstverständlich lassen sich hier auch Grenzwerte wie 07 (Wochentag) programmieren.

Es sei erwähnt, daß hier natürlich eine Verbindung zwischen PB 7 und NMI beim KIM-1 notwendig ist, damit der Timer den Interrupt auslösen kann. Er wird dann in der Interrupt-Routine selbst wieder neu gestartet. Da der 1-MHz-Quarz etwas neben der Frequenz liegen kann, ist mit dem Inhalt der Adresse 03DC eine Feinjustierung der Uhr möglich. *Herwig Feichtinger*

Sortierprogramm für den Z 80

Das nachstehend beschriebene Programm sortiert einen bis zu 64 KByte großen Datenbereich in absteigender Reihenfolge der Zelleninhalte.

Das Programm wurde auf dem Z-80-KIT realisiert. Die Anfangsadresse muß in das Doppelregister HL und die Endadresse der Daten muß in das Doppelregister DE eingegeben werden. Nach der Eingabe und nach Drücken der Starttaste prüft das Programm (*Bild*), ob die Endadresse größer als die Anfangsadresse ist. Wird diese Frage verneint, springt das Programm ins Monitorprogramm des KIT und ist somit beendet. Wird die Frage bejaht, sucht das Programm im Speicherbereich von HL bis DE den größten Wert heraus und speichert ihn auf die Anfangsadresse des Datenbereiches ab. Danach wird die Anfangsadresse um eins erhöht, und ein neuer Größtwert wird in diesem um eins verkürzten Datenbereich gesucht. Ist er gefunden, so wird dieser wieder um eins erhöht. Das



Flußdiagramm des Sortierprogramms. Im Registerpaar HL steht die Anfangsadresse des Datenblockes, in DE die Endadresse; St ist das Stack-Register, A der Akkumulator und unter (HL) verstehen wir den durch HL adressierten Speicherplatz

geschieht so lange, bis die Anfangsadresse gleich der Endadresse ist und das Programm somit beendet ist. Die *Tabelle* zeigt den Z-80-Maschinencode.

Die Realisierung der aus dem Flußdiagramm ersichtlichen Doppelregisterabfragen erfolgt mit einem Sprung in ein Unterprogramm, in dem die Differenz von HL und DE gebildet wird. Nach Rücksprung aus diesem Unterprogramm erfolgt die Frage, ob diese Differenz Null bzw. größer Null ist. Die dadurch bedingten Sprünge erfolgen indirekt, d. h. es wird zum Befehlszähler der Sprungabstand hinzugezählt (positiv bzw. negativ). Durch diese „indirekte Adressierung" ist es leicht möglich, das Programm in einen anderen Speicherbereich zu schreiben; man muß dann nur die zwei Unterprogrammaufrufe entsprechend ändern (3C50, 3C59).

Der Datenaustausch zwischen dem Akkumulator A und dem Speicherplatz (HL) erfolgt durch Laden von Register C mit dem Wert aus (HL) und Abspeichern von dem Wert aus A in den Platz (HL); danach wird A mit dem Wert aus C geladen.

Die Beendigung des Programms wird durch einen Sprung in das Monitorprogramm des KIT erreicht, wodurch der Inhalt von HL im Adressendisplay angezeigt wird. Bei Verwendung des Sortierprogramms als Unterprogramm muß dieser Sprung in einen Return-Befehl umgeändert werden (3C73).

Jörg Ullrich

Tabelle: Maschinencode für das Sortierprogramm

Stack-Pointer 3CFFH
Startadresse 3C50H

3C50	CD 6C 3C
3C53	28 1E
3C55	30 1C
3C57	E5
3C58	7E
3C59	CD 6C 3C
3C5C	28 09
3C5E	23
3C5F	BE
3C60	30 F7
3C62	4E
3C63	77
3C64	79
3C65	18 F2
3C67	E1
3C68	77
3C69	23
3C6A	18 E4
3C6C	E5
3C6D	37
3C6E	3F
3C6F	ED 52
3C71	E1
3C72	C9
3C73	C3 0D 01

Hans Werner

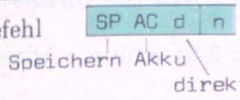
Mikrocomputer

Eine kleine Einführung

18. Teil

Speichern

Der Befehl



bewirkt: Speichere den Inhalt des Akkumulators in Speicherzelle n. Die Befehlszyklen LA und SP sind fast identisch.

Unterschiede:

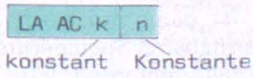
1. Datenpfeil 4 muß vom Akku zum Speicher zeigen.
2. Während Zyklus 4 erscheint kein L/S-Impuls, denn es wird nicht aus dem Speicher ausgelesen (gestrichelte Impulsreihe).
3. Speicheradresse 9 ist unzulässig. Warum?

Lösung:
Die Adressen 0...255 sprechen das ROM an, und aus diesem kann nur ausgelesen werden.

Laden einer Konstanten

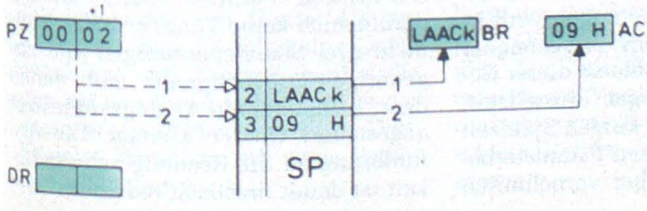
Hinter dem Operationsteil der beiden ersten Befehle steht direkt die Speicheradresse. Der Adreßteil kann aber auch eine andere Bedeutung haben.

Durch den Befehl

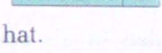


wird nicht der Inhalt der Speicherzelle n, sondern die Konstante n in den Akkumulator geladen. Die Konstante darf die Zahl 255 (1 Byte) nicht überschreiten.

Solche Befehle benötigen bei unserem Modell 2 Zyklen und damit nur die halbe Bearbeitungszeit.



Das Datenregister wird nicht angesprochen. Es ist leicht einzusehen, daß die Anweisung



keinen Sinn hat.

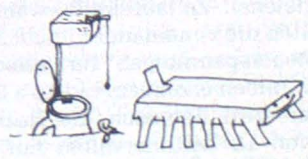
Training

Was bedeuten die Befehle?

1. SP AC d 400 D _____
2. LA AC k 20 D _____
3. LA AC d 512 D _____
4. SP AC d 100 D _____
5. SP AC d 512 D _____

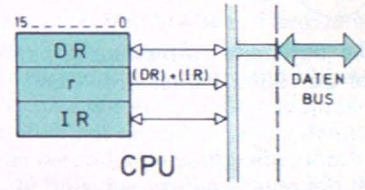
Hilfe:
Mit 8 bit lassen sich die Dezimalzahlen 0...255, mit 9 bit die Zahlen 0...511 darstellen. D steht für Dezimalzahl.

Lösungen:
1. Speichere den Inhalt des Akkus in Speicherzelle 400 des RAM: (AC) nach 400
2. Lade die Zahl 20 in den Akku: 20 in AC
3. Eingabe: Inhalt des Eingaberegisters in den Akku laden: (ER) in AC
4. Geht nicht! Adressen von 0...255 sprechen das ROM an.
5. Ausgabe: der Inhalt des Akkus wird in das Ausgaberegister gespeichert: (AC) nach AR



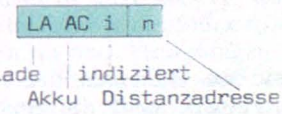
Index-Register

Der Adreßteil eines Befehls kann noch eine weitere Bedeutung haben. Vorher müssen wir allerdings die CPU geringfügig erweitern.

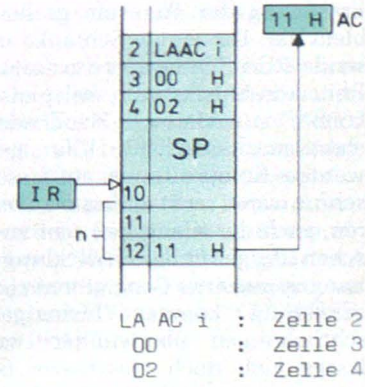


Hinzu kommt das Indexregister, manchmal auch Datenzähler genannt. Ein spezieller Adreßrechner r addiert den Inhalt des Datenregisters zum Inhalt des Indexregisters und gibt die Summe als neue Adresse auf den Datenbus.
Ausgelöst wird der Vorgang durch sogenannte indizierte Befehle. Sie erhalten als Kennzeichen ein i.

Als Beispiel soll der Befehl



dienen.



LA AC i : Zelle 2
00 : Zelle 3
02 : Zelle 4

Im Indexregister sei ein Wert von 10 (dezimal) gespeichert. Zuerst wird der Operationsteil LAACi in das Befehlsregister geladen. Dann rechnet r die Summe (DR) + (IR) = 12 aus, und der Inhalt der adressierten Speicherzelle 12 gelangt in den Akkumulator.

IR kann als Zeiger aufgefaßt werden, der auf einen bestimmten Speicherplatz deutet. Die wirkliche Adresse errechnet sich aus:

```
  Inhalt von      IR
+ Distanzadresse  n
-------------------
= adressierte Speicherzelle
```

Indizierte Befehle sind natürlich sinnlos, wenn nicht der Wert des Indexregisters verändert werden kann. Solche Befehle sind selbstverständlich im Befehlsvorrat des Modells enthalten.

Die indizierte Adressierung vereinfacht in vielen Fällen die Arbeit des Programmierers und veringert die Anzahl der benötigten Speicherplätze. Wenn mehr Befehle bekannt sind, wird das an einem geeigneten Programm gezeigt. Vorläufig beschränken wir uns auf eine kurze Wiederholung. Als Beispiel dient die Addition AD.

Direkte Adressierung

Struktur:

Operation	H-Adresse	T-Adresse
1. Byte	2. Byte	3. Byte

Beispiel:

AD d	00 H	07 H

Addiere den Inhalt des Akkumulators zum Inhalt der Speicherzelle 07 H (H = hexadezimal). Die Summe steht nach der Rechnung im Akkumulator. Kurzschreibweise: (AC)+(07) in AC

AC symbolisiert den Akku, (AC) seinen Inhalt.

Im Adreßteil des Befehls steht direkt die Adresse der gewünschten Speicherzelle.

Bei manchen Mikrocomputern wird zwischen kurzen und erweiterten Befehlen unterschieden:

kurz: 1 Byte für Adresse
erweitert (extended): 2 Byte für die Adresse

Allgemeine Aussagen über die Befehlsstruktur können nicht gemacht werden. Sie ist vom Fabrikat abhängig.

(Fortsetzung folgt)

Computer als Komponisten-Helfer

Der im folgenden beschriebene Sequenzer für spannungsgesteuerte Synthesizer mit dem PACE-Mikroprozessor von National Semiconductor eignet sich zum Komponieren, Modifizieren und Realisieren eines dreistimmigen Tonsatzes praktisch beliebiger Länge; eine recht interessante Anregung auch für musikalische Hobby-Programmierer.

Seit den Anfängen der elektronischen Musik stellt die Ansteuerung mehrerer elektronischer Elemente zur Erzeugung der Töne ein großes Problem dar. Der einige Schränke umfassende RCA-Synthesizer im Columbia-Princeton-Musikstudio beispielsweise konnte entweder von Hand oder mit elektromechanischer Hilfe gespielt werden. Kompositionen auf dieser Maschine waren recht mühsam zu realisieren, auch der Klang ließ sehr zu wünschen übrig. Mit der Enwicklung spannungsgesteuerter Generatoren zur Tonerzeugung konnten bisherige Beschränkungen überwunden werden, lassen sich doch elektrische Steuerspannungen mit Hilfe von Tastenfeldern, Lochstreifen oder vorprogrammierten Elementen wie Abtasthaltegliedern und Sequenzern leicht in Echtzeit realisieren. Insbesondere die programmierbaren Steuerspannungsquellen erlauben eine Vorbereitung, welche nicht in Echtzeit ablaufen muß. Trotzdem bleibt das Problem der Parametereinstellung (Tonhöhe, Tondauer) bestehen. Das Hauptproblem hat sich nur von der Echtzeit-Forderung auf das physische Einstellen von Knöpfen und Schaltern verlagert.

Der analoge Sequenzer ist das heute am weitesten verbreitete vorprogrammierbare Gerät. Vom Aufbau her besteht er im Prinzip aus einer Matrix von Potentiometern: Die Spalten gehören zum gleichen Zeitabschnitt, während die Zeilen die voneinander unabhängigen Steuerspannungen für verschiedene Stimmen erzeugen. Ein 16 x 3-Sequenzer wird demnach die Realisierung von 16 Zeitintervallen mit drei Stimmen ermöglichen. Zusätzlich zu diesen drei Steuerspannungen wird in jedem Zeitintervall ein Triggersignal erzeugt. Die Hauptprobleme dieser Geräte liegen in der langen Einstellzeit, einer verhältnismäßig kurzen Spielzeit und einem beschränkten Parameterbereich: Sie werden daher vornehmlich zur Erzeugung immer wiederkehrender Tonfolgen eingesetzt.

Mit dem Einsatz des Digitalrechners können die Probleme des analogen Sequenzers gelöst werden: Die Sequenzlänge ist dann nur noch eine Frage des verfügbaren Speicherumfangs, und die Programmierung wird einfacher. Trotzdem haben sich Sequenzer mit Digitalrechnern wegen der hohen Kosten nicht einführen können. Der Grund dafür lag darin, daß der Rechner außer der Abspielsteuerung noch die Töne erzeugte. Dafür benötigt man aber einen schnellen und damit teuren Rechner.

Der digitale Sequenzer, welcher hier beschrieben werden soll, erzeugt darum auch keine Töne. Er liefert vielmehr drei Steuerspannungen und zugehörige Triggersignale, mit denen dann kommerzielle Analogsynthesizer angesteuert werden können. Die Anforderung an die Rechengeschwindigkeit ist damit drastisch reduziert.

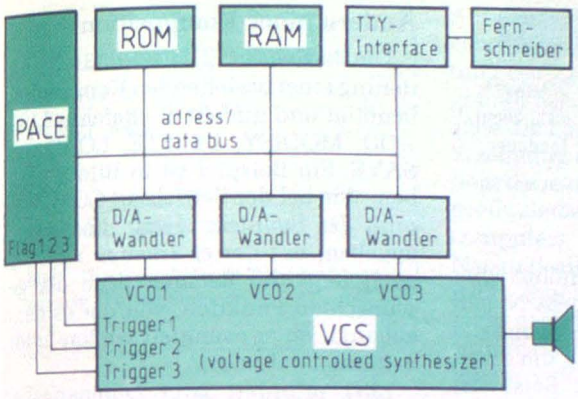


Bild 1. Blockschaltbild des mikrocomputergesteuerten Sequenzers

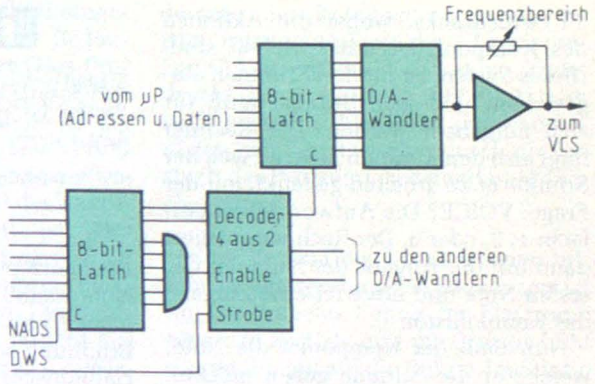


Bild 2. Die Ansteuerung der Digital/Analog-Wandler

Hardwareentwurf

Für dieses Projekt wurde der PACE von National Semiconductor gewählt. Dies ist ein 16-bit-Mikroprozessor mit einer typischen Befehlsausführungszeit von 8 Mikrosekunden und einem minicomputerähnlichen Befehlssatz. In einem Datenwort kann der vollständige Parametersatz einer Note in nur einer Speicherzelle untergebracht werden. Dies ist ein großer Vorteil gegenüber 8-bit-Prozessoren, da hier nur ein Datentransfer pro Ausgangsperiode statt deren zwei benötigt wird.

An den Datenbus des Systems, das in *Bild 1* dargestellt ist, sind noch ein Festwertspeicher (ROM) als Speicher für das Betriebsprogramm, ein Schreib-Lesespeicher (RAM) für die Aufbewahrung der abzuspeichernden Sequenz, welche in Form einer Liste im Speicher steht, eine Fernschreiberschnittstelle für den Dateneingang und den Dialog zwischen Komponist und Mikrocomputer sowie drei Digital/Analog-Wandler für die Ansteuerung der Synthesizer angeschlossen.

Die Befehlsverarbeitung soll nun am Beispiel der Digital/Analog-Wandlung demonstriert werden. Die dazu notwendige Schaltung ist im *Bild 2* dargestellt, während *Bild 3* den zeitlichen Ablauf verdeutlicht.

Zuerst gibt der Prozessor die Adresse des anzusteuernden Wandlers auf den Adreßbus. Mit Hilfe des NADS-Pulses wird die Adresse in einen Zwischenspeicher eingegeben. Der Decoder hinter diesem Speicher (in Bild 2 unten) wählt einen der drei Digital/Analog-Wandler an, der die folgenden Daten übernehmen soll. Danach werden die Daten auf den Bus gegeben und in das Ausgangsregister des jeweiligen Digital/Analog-Wandlers mit dem Datenpuls DWS geladen.

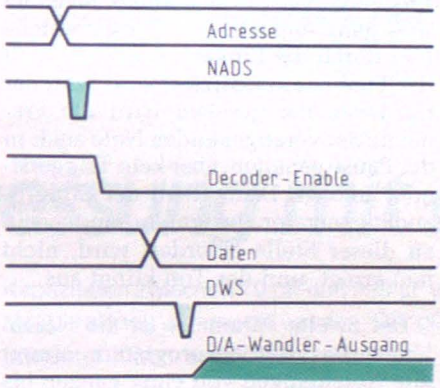


Bild 3. Zeitlicher Ablauf der Ansteuerung für die Digital/Analog-Wandler

Wichtig für den Systementwurf ist jetzt die Festlegung des Tonumfangs, über den ein angeschlossener Synthesizer gesteuert werden soll. Ein 8-bit-Digital/Analog-Wandler erzeugt 256 Spannungsstufen, deren Höhe vom Endwertpotentiometer des Wandlers eingestellt werden kann. Beim ausgeführten Beispiel wurde ein Bereich von 64 Tönen gewählt, da die Auflösung zwischen zwei Tönen 4 Stufen beträgt und damit Vierteltöne und Vibratoeffekte ermöglicht.

Gleichzeitig mit dem Gleichspannungsausgangssignal der Digital/Analog-Wandler werden vom Prozessor Triggersignale erzeugt, welche an den unabhängig steuerbaren FLAG-Ausgängen des PACE-Mikroprozessors anstehen. Jedesmal, wenn ein neuer Ton erzeugt werden soll, erfolgt eine Anhebung des Pegels an den FLAG-Ausgängen. Die Triggersignale beeinflussen über den Einhüllendengenerator einen steuerbaren Verstärker, welcher damit die für die verschiedenen Stimmen charakteristischen Einschwingvorgänge erzeugt.

Softwareentwurf

Das eben beschriebene Hardwaresystem wird durch ein spezielles Betriebsprogramm auf tonschöpferische Belange ausgerichtet. Gerade im Hinblick auf den Musiker als Benutzer ist es besonders wichtig, eine ihm vertraute Sprache für den Dialog mit dem Rechner zu wählen statt einer obskuren, den Informatiker begeisternden Rechnersprache. Die Benutzerfreundlichkeit wird noch gesteigert durch die Fähigkeit des Systems, den Komponisten in englischem Klartext durch seine Komposition zu geleiten.

Das Komponieren erfolgt schrittweise zwischen den drei Phasen: Komponieren, Ändern und Spielen. In der Phase des Komponierens bringt der Komponist Ideen zu Papier, welche bislang noch nicht dokumentiert waren. Als nächstes wird er das Stück spielen und daraufhin noch Änderungen vornehmen. Die Reihenfolge der beiden letzten Phasen ist vertauschbar, und es hängt vom persönlichen Arbeitsstil des Komponisten ab, wie er die Reihenfolge wählt.

In diesem rechnergestützten System gibt es drei korrespondierende Betriebsarten: Komponieren (COMPOSE), Änderung (EDIT) und Ausführung (PLAY). Statt einer Systembeschreibung soll am praktischen Beispiel verfolgt werden, wie eine neue Komposition entsteht. Zu Programmbeginn fragt der Mikrocomputer den Komponisten nach seinen Wünschen: COMPOSE, EDIT or PLAY?

Komponieren

Angenommen, Polyhymniens Musenkuß streifte des Meisters Stirn, und es drängt ihn, seine Inspiration auf fünf Linien darzustellen; dann antwortet er mit COMPOSE oder einfach mit C. *Bild 4* zeigt den Ablauf anhand des

TTY-Ausdrucks, wobei die Aktionen des Komponisten unterstrichen sind. Dieses System ist für drei Stimmen eingerichtet, und jede Stimme muß für sich aufgebaut werden. Der Rechner fragt also den Komponisten, an welcher Stimme er zu arbeiten gedenkt, mit der Frage: VOICE? Die Antwort lautet einfach: 1, 2, oder 3. Der Rechner quittiert dann mit der Angabe der Nummer der ersten Note und erwartet eine Eingabe des Komponisten.

Nun muß der Komponist die Note, welche er der Stimme geben möchte, genau spezifizieren. Dazu wird ein geordnetes Paar von Daten eingegeben, welches die beiden Parameter, Tonhöhe und Dauer, festlegt. Dazu gibt es mehrere Möglichkeiten, aus denen diejenige gewählt wurde, welche die Tonhöhe durch den Notennamen spezifiziert.

Es erscheint etwas abwegig, bei einem analogen Synthesizer so zu verfahren, bewirkt doch eine Drehung am Stimmknopf des VCO etwa für ein gis die Lage irgendwo zwischen Infra- und Ultraschall. Die nächstbeste Methode, nämlich die Festlegung der Tonhöhe durch Intervalle – eine Methode, welche schon im Mittelalter angewendet wurde – bedeutet, daß die Notenliste unabhängig von der Frequenz wird und ausgehend von der ersten Note über den gesamten Frequenzbereich transponiert werden kann. Diese Methode ist aber für den Komponisten unpraktisch, denn es ist einfacher, mit Notennamen als mit Intervallen zu komponieren. Die

```
COMPUTER COMPOSTION PROGRAM
VERSION = 3
COMPOSE, EDIT OR PLAY? C
VOICE? 1
0001  1  A3,8
0002     C+3,4
0003     E4,16
0005     E1,32
0006     D5,2
0007     F2,4
0008     REST,6
0009     A+4,16
0010     STOP
COMPOSE, EDIT OR PLAY? 1
END
```

Bild 4. Beispiel für den Dialog mit dem Rechner während des Komponierens

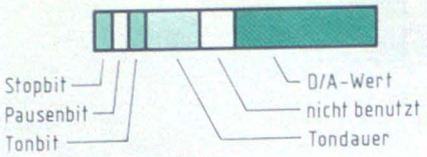


Bild 5. Wortformat der Notenspezifikation

Frequenzspezifikation (Tonhöhe) einer Note umfaßt drei Kenngrößen: Notenname (A bis G), Oktave (1 bis 5) und die Erhöhung oder Erniedrigung um einen Halbtonschritt (+ oder –). Beispielsweise ist das cis über dem Kammerton a mit 440 Hz ein C + 3.

Ein Spezialfall der Frequenz ist die Frequenz Null. In der Musik bedeutet dies ganz einfach eine Pause, welche hier durch die Eingabe von REST statt der Tonhöhe spezifiziert wird. Von der Hardware her gesehen wird die Frequenz der vorangehenden Note auch in der Pause gehalten, aber kein Triggersignal erzeugt. Damit wird der Einhüllendengenerator, dessen Vorhandensein an dieser Stelle gefordert wird, nicht neu erregt, und der Ton klingt aus.

Der zweite Parameter ist die Notenlänge. Das Betriebsprogramm nimmt alle Notenlängen von einer ganzen bis zu einer $^1/_{32}$-Note an.

Die Dauer wird einfach durch den Kehrwert, also durch eine 1 für eine ganze Note, durch 2 für eine halbe Note usw. programmiert. Verlängerte Noten lassen sich ganz analog zum gewohnten Notenbild durch Anhängen eines Punktes bezeichnen.

Ist die Notenliste für eine Stimme fertig, dann gibt der Komponist einfach STOP ein. Dies bewirkt einmal einen Rücksprung des Programms an den Anfang zur Frage nach der gewünschten Funktion und zusätzlich das Einfügen einer Kennung des Endes der Liste.

Die gesamte Information über die Art, Dauer und Frequenz einer Note ist in nur einem Datenwort der Länge 16 bit zusammengefaßt, dessen Format in *Bild 5* gezeigt wird. Links liegt das STOP-Bit, welches das Ende einer Komposition anzeigt. Das nächste ist das Pausenbit zur Unterdrückung des Triggers zu Beginn einer Pause. Das folgende Bit bezeichnet die Verlängerung einer Note um 50 %. Die Dauer ist in den nächsten drei bit enthalten und reicht von einer $^1/_{32}$-Note bis zu einer ganzen Note. Die nächsten beiden Stellen werden nicht benutzt, während die letzten acht der Aussteuerung der Digital/Analog-Wandler dienen.

Ändern einer Komposition

Die Betriebsart EDIT wird für die Änderung einer bestehenden Komposition benötigt und umfaßt die Befehle: LIST, ADD, MODIFY, DELETE, LOAD und SAVE. Ein Beispiel ist in Bild 6 gegeben. Wie bei der Betriebsart COMPOSE muß der Benutzer zuerst die Stimme angeben, mit der er arbeiten will. Danach fragt der Rechner nach der gewünschten Funktion, von der es dann abhängt, ob er eventuell weitere Informationen anfordert.

LIST erfordert zwei Zeilenangaben über Anfang und Ende; ADD, MODIFY und DELETE dagegen benötigen die Angabe nur einer Zeilenzahl. Die ADD-Funktion fügt eine Zeile nach der spezifizierten Zeile ein und schiebt jede folgende Zeile der Notenliste um eine Stelle höher.

Die letzten beiden Funktionen dieser Betriebsart sind SAVE und LOAD, mit denen die Speicherung einer Komposition für spätere Verwendung möglich ist. Einer der größten Vorteile des mikrocomputergesteuerten Sequenzers ist ja, daß die Komposition in einer speicherfähigen Form vorliegt. Als Speichermedium dient hier der Lochstreifen, in welchen mit dem SAVE-Befehl alle drei Stimmen gestanzt werden können.

Spielen

Der Komponist verfügt letztlich noch über die Betriebsart PLAY. Bevor der Mikrocomputer allerdings ein Stück über die drei angeschlossenen Synthesizer wiedergeben läßt, benötigt er noch zwei Informationen: Erstens die Anzahl der Wiederholungen (von 0 bis 9999)

```
COMPOSE, EDIT OR PLAY?  E
VOICE? 1
FUNCTION? DELETE
LINE NUMBER? 6
FUNCTION? MODIFY
LINE NUMBER? 5
0005 E+1,16
FUNCTION? LIST
LINE NUMBER? 4,7
0004 G-,2
0005 E+1,16
0006 F2,4
0007 REST,8
FUNCTION?
```

Bild 6. Dialog während der Änderungsphase

```
COMPOSE, EDIT OR PLAY? PLAY
REPEAT? 8
TEMPO? 44
GO? _
STOP
COMPOSE, EDIT OR PLAY? 1
```

Bild 7. Damit spielt der Rechner ein Largo mit acht Wiederholungen

und zweitens das Tempo, wie dies in *Bild 7* gezeigt wird. Das Tempo kann nun grundsätzlich von der Hardware als auch von der Software festgelegt werden: Hier wird der zweite Weg beschritten. Eine Tempozahl wird als Antwort auf die Frage: TEMPO? eingegeben und entspricht in ihrer Bedeutung der Zahl am Metronom. Das Programm beeinflußt aufgrund dieser Angabe die Länge der Zählschleife für die Erzeugung der Notenlänge. Der Vorteil liegt darin, daß das Tempo jederzeit reproduzierbar ist, da der Takt des Mikrocomputers quarzstabilisiert ist. Der Nachteil allerdings besteht darin, daß nach Spielbeginn das Tempo nicht mehr geändert werden kann. Die Hardwarerealisierung hingegen besteht aus einem externen einstellbaren Oszillator, mit dem die Notenlänge festgelegt wird.

Nachdem nun die Anzahl der Wiederholungen und die Angabe des Tempos eingegeben wurde, fragt der PACE an, ob er spielen darf: GO? Für die Wiedergabe der Komposition genügt es jetzt, irgendeine Taste der TTY niederzudrücken. Ist das Konzert zu Ende, druckt der Rechner STOP und fragt nach weiteren Wünschen des Komponisten, der bereits vorher durch die Taste (!) das Programm jederzeit unterbrechen kann.

Eine Version des beschriebenen mikrocomputergesteuerten Sequenzers ist im Los Angeles Center for Electronic Music in Betrieb, und die Gruppe LEM nimmt ihn mit auf die Bühne. Im Studio hat sich das Gerät gut bewährt, aber auf der Bühne ist die benutzte ASR 33 Teletype doch zu langsam. Abhilfe soll hier ein Bildschirmterminal und die Floppy Disk als Speicher für Kompositionen bringen.

Klaus Mezger

Kapazitätsmessung mit dem KIM-1

Kapazitäten zwischen 10 pF und 999 µF können mit vierstelliger Digitalanzeige in fünf Meßbereichen untersucht werden. Der günstigste Meßbereich wird dabei vom Mikrocomputer KIM-1 automatisch ermittelt.

Das Meßprinzip

Der zu messende Kondensator wird mit konstantem Strom bis auf eine Schwellenspannung U_s aufgeladen. Die dazu benötigte Zeit ist der Kapazität direkt proportional (*Bild 1*).

Der Computer hat nun die Aufgabe, die Zeit T_S zu messen. Hat die Kondensatorspannung nach der Zeit T_{max} die Schwelle U_s nicht erreicht, muß die Messung mit erhöhtem Strom wiederholt werden. Das beschriebene Programm erhöht den Strom dekadisch maximal viermal, dann wird Überlauf angezeigt (FF).

Der Computer erzeugt mit seinem internen programmierbaren Timer periodische Interrupt-Signale mit der Periodenzeit T_M. Diese Impulse beginnen bei $t = 0$ und werden gezählt bis $t = T_S$. Die Anzahl der Impulse ist proportional zur Zeit T_S und damit auch zur Kapazität C_X. Das Programm bewirkt das Lesen des Tastenfelds (Startbefehl), die Ansteuerung des Displays, die Steuerung des externen Zeitglieds NE 555, die Zeitmessung und die automatische Meßbereichs-Umschaltung.

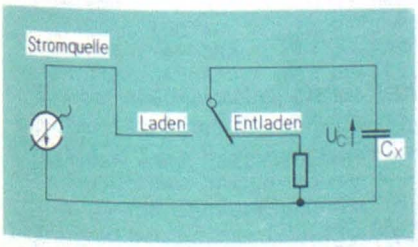


Bild 1. Die Kapazitätsmessung geschieht hier durch Messen der Zeit, die benötigt wird, um C mit einer Konstantstromquelle bis auf eine bestimmte Spannung aufzuladen

Das Programm

Das Hauptprogramm verwendet zum Lesen der KIM-Taste „0" (Start-Befehl) das im KIM-ROM gespeicherte Unterprogramm GETKEY und zur Ansteuerung des Displays SCANDS. Die externe Elektronik wird über die E/A-Leitungen PA 0...PA 7 und PB 7 vom Programm gesteuert:

PA 0...4:	Meßbereiche 1...5
PA 5:	Startimpuls für den Timer NE 555
PA 6:	Reset des NE 555
PA 7:	Ausgang des Zeitgliedes NE 555
PB 7:	Ausgang des KIM-Timers mit NMI verbinden.

Die periodischen Zählimpulse werden mit dem internen Timer (Adr. 170D) erzeugt, welcher bei seinem Nulldurchgang ein Interruptsignal an PB 7 abgibt und anschließend in der Interrupt-Routine wieder gestartet wird, bis die Messung durch ein Ausgangs-Signal des NE 555 beendet wird.

Das KIM-Display soll direkt die gemessene Kapazität anzeigen. Deshalb müssen T_M, T_S und der Strom I_C in einem festen Verhältnis zueinander stehen.

T_M, die Periodenzeit der Zählimpulse, setzt sich aus folgenden Zeiten zusammen:

48 µs	KIM-Timer
+ 18 µs	Zeit vom Start des Interrupts bis zum Timer-Start
+ ca. 4 µs	mittlere Wartezeit bis zum Interrupt-Service durch die CPU

T_M ist also rund 70 µs. Damit wird $T_{max} = 9999 \cdot 70\ \mu s \approx 700$ ms.

Wenn vom KIM-Timer ein Interrupt-Signal abgegeben wird, aber kein Signal vom NE 555 vorliegt, inkrementiert das Programm den Zähler, und im Falle eines Überlaufs (>9999) wird der Speicher OVERFL = 1 gesetzt. Außerdem wird der KIM-Timer wieder gestartet.

Wird dagegen ein Signal vom NE 555 an PA 7 erkannt, erfolgt ein direkter Rücksprung zum Hauptprogramm.

Die externe Elektronik

Bild 2 zeigt die extern benötigte Hardware. Die Transistoren T_3 und T_4 bilden mit R_5, R_6 und R_{EX} eine Konstantstromquelle mit hohem Innenwiderstand. D_2 und D_3 kompensieren die Temperaturdrift der Basis-Emitter

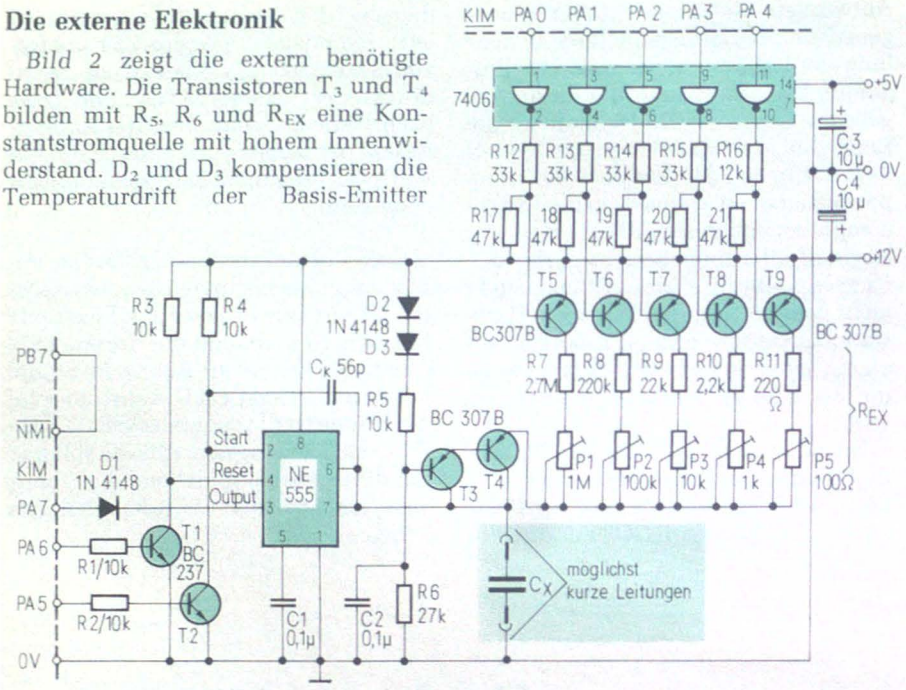


Bild 2. Schaltbild der zur Kapazitätsmessung benötigten Hardware

Spannungen U_{BE3} und U_{BE4}. Das Zeitglied NE 555 arbeitet als Schwellwertschalter und besorgt nach erfolgter Messung die Entladung von C_X. Der Kondensator C_X kann auch durch den Computer entladen werden, indem dieser den Reset Eingang über T_1 an Masse legt.

Der Kondensator C_K ‚kompensiert' die zu C_X parallel liegenden parasitären Kapazitäten. (Ohne C_K wird auch bei offenen Meßklemmen etwa 30 pF angezeigt). T_5 bis T_9 und der TTL-Treiber 7406 schalten den entsprechenden Emitterwiderstand der Konstantstromquelle zu. Da beim KIM ohnehin schon eine 5-V-Betriebsspannung vorhanden ist, kann man mit dem TTL-Treiber fünf Transistoren einsparen.

Mit den Trimmern P_1...P_5 wird die Schaltung abgeglichen, wobei am besten schon anderweitig ausgemessene Kondensatoren verwendet werden. Es ergeben sich mit der angegebenen Dimensionierung folgende Meßbereiche:

Bereich 1: 10 pF...99,99 nF
Bereich 2: 100 nF...999,9 nF
Bereich 3: 1 µF...9,999 µF
Bereich 4: 10 µF...99,99 µF
Bereich 5: 100 µF...999,9 µF

Inbetriebnahme

Das KIM-Adressen-Display soll die Kapazität anzeigen, das Daten-Display den Meßbereich. Das Programm wird zunächst ohne Elektronikteil getestet; PB 7 muß jedoch mit dem NMI-Eingang verbunden sein.

Nach dem Programmstart muß in der Anzeige 0000 01 zu sehen sein. Jetzt wird die Taste 0 gedrückt und die Kapazitätsanzeige zählt während etwa 0,7s auf 9999 hoch; dann wird auf Bereich 2 umgeschaltet und das Spiel wiederholt sich, bis der Bereich 5 durchlaufen ist. Dann wird auf dem Daten-Display ‚FF', d. h. Überlauf, angezeigt. Der Zähler bleibt nicht stehen, da die Schaltung noch nicht angeschlossen ist. Läuft das Programm nicht, prüfe man zuerst, ob der Interrupt-Vektor geladen ist (17FA: 80, 17FB: 02). Läuft das Programm, so wird die Elektronik angeschlossen, wobei die Betriebsspannungen dem KIM-Netzteil entnommen werden können. Man prüfe insbesondere die richtige Polarität der Diode D1.

Wenn kein C_X angeschlossen ist, sollte das Display 0000 01 anzeigen. Wird eine kleine Kapazität angezeigt (z. B. 10 pF), sind die Anschlußdrähte zu den Meßklemmen zu lang oder C_K zu klein (C_K sollte jedoch kleiner als 68 pF bleiben). Der Abgleich erfolgt am besten erst nach ca. 5 min. Betriebszeit und im Bereich 1 mit einem bekannten Kondensator. Am oberen Bereichsende schwankt die Anzeige von Messung zu Messung um ca. ± 30 pF. Das ist jedoch nur ± 0,03 % Meßunsicherheit und rührt von der Toleranz der Zeit T_M her.

Mit Vorteil wird die ganze externe Schaltung über einen Stecker mit dem KIM-1 verbunden, der so ausgelegt sein sollte, daß beliebige andere Geräte auch angeschlossen werden können. Der Meßfehler hängt in erster Linie von der Genauigkeit des Abgleichs ab. Die Schaltung ermöglicht Fehler <1 %. ■

```
0200 A9 7F     LDA #7F   I/O-PORTS
0202 8D 01 17  STA 1701  SETZEN
0205 A9 01     LDA #01
0207 8D 00 17  STA 1700  MESSBER.1
020A 85 F9     STA F9
020C A9 00     LDA #00   ZAEHLER
020E 85 FA     STA FA    AUF NULL
0210 85 FB     STA FB    SETZEN
0212 20 1F 1F  JSR 1F1F  ANZEIGE
0215 20 6A 1F  JSR 1F6A  NULL-TASTE
0218 C9 00     CMP #00   ABFRAGEN
021A D0 F6     BNE 0212
021C A9 01     LDA #01   MESSBER.1
021E 8D 00 17  STA 1700
0221 85 F9     STA F9
0223 A9 00     LDA #00   ZAEHLER UND
0225 85 FA     STA FA    UEBERLAUF
0227 85 FB     STA FB    AUF NULL
0229 85 00     STA 00    SETZEN
022B A9 01     LDA #01
022D 8D 0E 17  STA 170E  TIMER-START
0230 AD 00 17  LDA 1700
0233 09 20     ORA #20   STARTIMPULS
0235 8D 00 17  STA 1700  FUER NE 555
0238 29 9F     AND #9F   START- UND
023A 8D 00 17  STA 1700  RESETIMPULS
023D 20 1F 1F  JSR 1F1F  ANZEIGE-SUBR
0240 AD 00 17  LDA 1700  TEST, OB
0243 10 CD     BPL 0212  MESS-ENDE
0245 A5 00     LDA 00    UEBERLAUF-
0247 F0 F4     BEQ 023D  PRUEFUNG
0249 AD 00 17  LDA 1700
024C 09 40     ORA #40   NE 555
024E 8D 00 17  STA 1700  RUECKSETZEN
0251 0A        ASL A     TEST, OB
0252 0A        ASL A     BEREICH 5
0253 0A        ASL A     ERREICHT
0254 10 07     BPL 025D  NEIN
0256 A9 FF     LDA #FF   UEBERLAUF-
0258 85 F9     STA F9    ANZEIGE
025A 4C 12 02  JMP 0212
025D A9 FF     LDA #FF   RESET-SIGNAL
025F 8D 07 17  STA 1707  FUER NE 555
0262 AD 07 17  LDA 1707  VERLAENGERN
0265 F0 FB     BEQ 0262
0267 AD 00 17  LDA 1700
026A 0A        ASL A     NAECHSTE
026B 09 40     ORA #40   STROMSTUFE
026D 8D 00 17  STA 1700
0270 E6 F9     INC F9    BER.-ANZEIGE
0272 4C 23 02  JMP 0223

INTERRUPT-ROUTINE
0280 48        PHA
0281 AD 00 17  LDA 1700  INTERRUPT-
0284 10 17     BPL 029D  QUELLE TESTE
0286 A9 06     LDA #06
0288 8D 0D 17  STA 170D  TIMER-START
028B F8        SED
028C 18        CLC       DEZIMALE
028D A9 01     LDA #01   2-BYTE-
028F 65 FA     ADC FA    ADDITION
0291 85 FA     STA FA    MIT 01
0293 A9 00     LDA #00
0295 65 FB     ADC FB
0297 85 FB     STA FB    UEBERLAUF
0299 90 05     BCC 02A0  PRUEFEN
029B E6 00     INC 00
029D AD 06 17  LDA 1706
02A0 D8        CLD
02A1 68        PLA
02A2 40        RTI
```

Bild 3. Das 6502-Programm, mit dem die Steuerung der Hardware aus Bild 2 erfolgt. Der NMI-Vektor ist auf die Adresse 0280 zu setzen

TI-59-Magnetkarten nicht austauschbar

Der programmierbare Taschenrechner TI-59 gehört heute sicher zu den am weitesten verbreiteten. Seine Benutzer müssen aber leider eine recht unangenehme Eigenschaft dieses kleinen Wunderwerks hinnehmen, die man wohl einer falschen Auffassung von Sparsamkeit bei seinen Entwicklern zuschreiben muß: Magnetkarten, die auf einem solchen Rechner bespielt wurden, werden von einem anderen TI-59 nicht unbedingt fehlerfrei gelesen. Dies wurde uns kürzlich von Texas Instruments bestätigt.

Das wäre ja noch nicht weiter schlimm – dann blinkt eben die Anzeige, so möchte man meinen, man weiß Bescheid, daß ein Fehler auftrat und behilft sich mit dem Abtippen des Programm-Listings. Aber nein! Denn der Hersteller hat leider ein recht schludriges Maschinenprogramm zur Prüfsummenberechnung implementiert, und wenn man „Glück" hat, schluckt der TI-59 etliche Lesefehler, ohne dies dem Benutzer mitzuteilen. Und der erstellt davon womöglich ein Listing...

Allerdings kann man dieser schludrigen Rechner-Software auch Gutes abgewinnen: Manche Leute versehen nämlich ihre Magnetkarten mit einem Schutz-Flag, das dem Rechner beim Einlesen des Programms u. a. verbietet, einen List-Befehl auszuführen. Wird dieses Schutzflag nun zufällig falsch gelesen (da kann man nachhelfen...), so läßt sich das Programm wieder auflisten, wenn auch Gefahr besteht, daß außer dem Schutzflag auch andere Programmteile falsch gelesen werden.

Übrigens gewährleistet Hewlett-Pakkard für seine Rechner HP-67 und HP-97 offiziell die Austauschbarkeit der Magnetkarten, läßt sich dies aber auch teuer bezahlen. Wesentlich für die Kompatibilität ist erstens die Übereinstimmung der Motorgeschwindigkeit und zweitens die exakte Tonkopf-Spaltstellung bei allen Rechnern. Bei Texas Instruments hat man den Kopf schlicht und einfach eingegossen – es könnte ja sonst jemand auf die Idee kommen, daran zu drehen! Fe.

µC-Nachrichten

Unter dem Titel **„The Best of Micro"** erschien jetzt eine Zusammenstellung von interessanten Programmen und Tips für 6502-Anwender, die Mikrocomputer wie KIM, PET, TIM oder Apple besitzen. Mehrere heraustrennbare Blätter enthalten eine Aufstellung der Monitorprogramme und der Speicheraufteilung der genannten Mikrocomputer. Für Maschinensprache-Spezialisten ist ein in BASIC geschriebener Disassembler abgedruckt. Das leider nur in Englisch erhältliche Buch ist von M. Nedela in Markdorf zu haben.

□

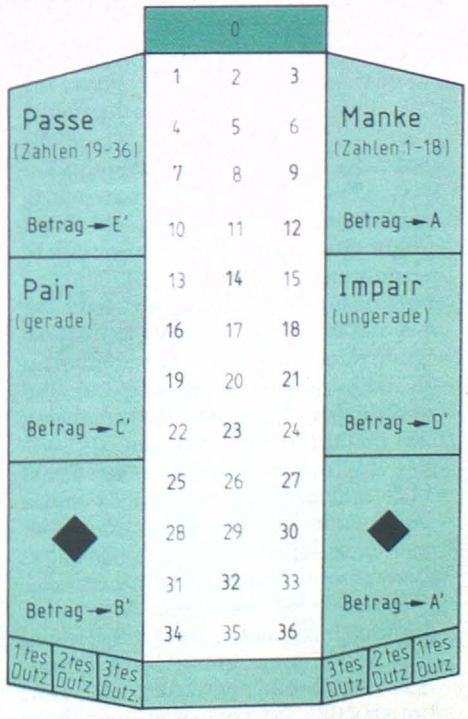
Intel hat ein **statisches 8-Kbit-RAM** eingeführt. Es trägt die Bezeichnung 8185 und ist 1 K x 8 bit organisiert. Die Byte-orientierten Speicherchips werden im 18-Pin-Gehäuse geliefert. Die Zugriffszeit beträgt 300 ns, auf Wunsch auch nur 200 ns; als Stromversorgung werden nur +5 V benötigt. Die Speicher werden ab Mitte 1979 lieferbar sein und sind 8085-kompatibel.

Fred. H. Heller

Roulett auf dem TI-59

Das hier vorgestellte Roulett-Programm ist weit mehr als ein einfacher Zufalls-Generator: Es berücksichtigt genau die Spielregeln des „richtigen" Roulett und verwaltet auch das Spielgeld.

Bei jedem Spiel können beliebige Einsätze in beliebigen Feldern gleichzeitig gesetzt werden. Die Einsätze werden vom Spielkapital abgebucht, die Gewinne dem Spielkapital hinzuaddiert.



für Zahlen setzen:
Betrag → D; Zahl → R/S

setzen fürs 1te, 2te u. 3te Dutzend:
Betrag → B; Dutzend → R/S

Bild 1. Spielplan für das TI-59-Roulett

Die Bedienung des Rechners ist denkbar einfach. Nach dem Eingeben des Programms bzw. dem Einlesen der Magnetkarte drückt man die Taste C, wartet etwa 10...60 s und hält dann die RST-Taste gedrückt, bis in der Anzeige eine Zahl erscheint. Nun gibt man das Anfangsguthaben ein und speichert es in Register 19 (STO 19).

Wenn alle Beträge nach dem Schema in *Bild 1* gesetzt sind, bringt man durch R/S die „Kugel ins Rollen". Eine Betätigung der Taste E zeigt jederzeit das vorhandene Spielkapital an.

Das Programm (*Bild 2*) erzeugt eine echte gleichmäßig verteilte Zufallszahl zwischen 0 und 36, berechnet die Gewinnquote, verwaltet das Spielkapital, zeigt durch eine blinkende Null an, daß der Spieler pleite ist und zahlt den Gewinn an die Spielkapital-Verwaltung aus. Nach jedem Spiel wird automatisch der Spielplan wieder „abgeräumt".

Und noch eines hat dieses Programm mit dem echten Roulett gemeinsam: Die Bank, sprich: der Rechner, gewinnt immer!

```
000  61 GTO
001  54  )
002  76 LBL
003  13  C
004  86 STF
005  01  01
006  36 PGM
007  15  15
008  10 E'
009  01  1
010  06  6
011  36 PGM
012  15  15
013  15  E
014  00  0
015  36 PGM
016  15  15
017  11  A
018  03  3
019  06  6
020  83  .
021  09  9
022  09  9
023  36 PGM
024  15  15
025  12  B
026  76 LBL
027  53  (
028  36 PGM
029  15  15
030  13  C
031  87 IFF
032  01  01
033  53  (
034  59 INT
035  42 STO
036  07  07
037  61 GTO
038  33 X²
039  76 LBL
040  54  )
041  61 GTO
042  53  (
043  76 LBL
044  16 A'
045  22 INV
046  44 SUM
047  19  19
048  65  ×
049  02  2
050  95  =
051  42 STO
052  59  59
053  25 CLR
054  91 R/S
055  61 GTO
056  53  (
057  76 LBL
058  17 B'
059  22 INV
060  44 SUM
061  19  19
062  65  ×
063  02  2
064  95  =
065  42 STO
066  58  58
067  25 CLR
068  91 R/S
069  61 GTO
070  53  (
071  76 LBL
072  18 C'
073  22 INV
074  44 SUM
075  19  19
076  65  ×
077  02  2
078  95  =
079  42 STO
080  15  15
081  25 CLR
082  91 R/S
083  61 GTO
084  53  (
085  76 LBL
086  19 D'
087  22 INV
088  44 SUM
089  19  19
090  65  ×
091  02  2
092  95  =
093  42 STO
094  14  14
095  25 CLR
096  91 R/S
097  61 GTO
098  53  (
099  76 LBL
100  10 E'
101  22 INV
102  44 SUM
103  19  19
104  65  ×
105  02  2
106  95  =
107  42 STO
108  16  16
109  25 CLR
110  91 R/S
111  61 GTO
112  53  (
113  76 LBL
114  11  A
115  22 INV
116  44 SUM
117  19  19
118  65  ×
119  02  2
120  95  =
121  42 STO
122  17  17
123  25 CLR
124  91 R/S
125  61 GTO
126  53  (
127  76 LBL
128  12  B
129  22 INV
130  44 SUM
131  19  19
132  65  ×
133  03  3
134  95  =
135  42 STO
136  18  18
137  03  3
138  32 X↕T
139  25 CLR
140  91 R/S
141  67  EQ
142  34 fX
143  85  +
144  01  1
145  95  =
146  67  EQ
147  35 1/X
148  43 RCL
149  18  18
150  42 STO
151  57  57
152  25 CLR
153  91 R/S
154  61 GTO
155  53  (
156  76 LBL
157  34 fX
158  43 RCL
159  18  18
160  42 STO
161  13  13
162  25 CLR
163  91 R/S
164  61 GTO
165  53  (
166  76 LBL
167  35 1/X
168  43 RCL
169  18  18
170  42 STO
171  12  12
172  25 CLR
173  91 R/S
174  61 GTO
175  53  (
176  76 LBL
177  14  D
178  22 INV
179  44 SUM
180  19  19
181  65  ×
182  03  3
183  06  6
184  95  =
185  42 STO
186  18  18
187  25 CLR
188  91 R/S
189  85  +
190  02  2
191  00  0
192  95  =
193  42 STO
194  00  00
195  43 RCL
196  18  18
197  72 ST*
198  00  00
199  25 CLR
200  91 R/S
201  61 GTO
202  53  (
203  76 LBL
204  15  E
205  43 RCL
206  19  19
207  91 R/S
208  43 RCL
209  20  20
210  91 R/S
211  61 GTO
212  17 B'
213  76 LBL
214  33 X²
215  76 LBL
216  33 X²
217  29 CP
218  67  EQ
219  43 RCL
220  32 X↕T
221  03  3
222  09  9
223  32 X↕T
225  21 2ND
226  23 LNX
227  62 PG*
228  39  39
229  76 LBL
230  21 2ND
231  74 SM*
232  23  23
233  22 INV
234  25 CLR
235  19 D'
236  26 2ND
237  09  9
238  66 PAU
239  94 +/-
240  67  EQ
241  65  ×
242  43 RCL
243  14  14
244  76 LBL
245  75  -
246  44 SUM
247  19  19
248  02  2
249  05  5
250  32 X↕T
251  43 RCL
252  07  07
253  77  GE
254  85  +
255  01  1
256  03  3
257  32 X↕T
258  43 RCL
259  07  07
260  77  GE
261  60 DEG
262  43 RCL
263  57  57
264  76 LBL
265  95  =
266  44 SUM
267  19  19
268  02  2
269  09  9
270  32 X↕T
271  43 RCL
272  07  07
273  77  GE
274  70 RAD
275  01  1
276  09  9
277  32 X↕T
278  43 RCL
279  07  07
280  77  GE
281  80 GRD
282  01  1
283  01  1
284  32 X↕T
285  43 RCL
286  07  07
287  77  GE
288  70 RAD
289  85  +
290  01  1
291  95  =
292  76 LBL
293  70 RAD
294  55  ÷
295  02  2
296  95  =
297  22 INV
298  59 INT
299  29 CP
300  67  EQ
301  90 LST
302  43 RCL
303  58  58
304  76 LBL
305  25 CLR
306  44 SUM
307  19  19
308  43 RCL
309  07  07
310  85  +
311  02  2
312  00  0
313  95  =
314  42 STO
315  00  00
316  73 RC*
317  00  00
318  76 LBL
319  38 SIN
320  44 SUM
321  19  19
322  00  0
323  42 STO
324  20  20
325  42 STO
326  21  21
327  42 STO
328  22  22
329  42 STO
330  23  23
331  42 STO
332  24  24
333  42 STO
334  25  25
335  42 STO
336  26  26
337  42 STO
338  27  27
339  42 STO
340  28  28
341  42 STO
342  29  29
343  42 STO
344  30  30
345  42 STO
346  31  31
347  42 STO
348  32  32
349  42 STO
350  33  33
351  42 STO
352  34  34
353  42 STO
354  35  35
355  42 STO
356  36  36
357  42 STO
358  37  37
359  42 STO
360  38  38
361  42 STO
362  39  39
363  42 STO
364  40  40
365  42 STO
366  41  41
367  42 STO
368  42  42
369  42 STO
370  43  43
371  42 STO
372  44  44
373  42 STO
374  45  45
375  42 STO
376  46  46
377  42 STO
378  47  47
379  42 STO
380  48  48
381  42 STO
382  49  49
383  42 STO
384  50  50
385  42 STO
386  51  51
387  42 STO
388  52  52
389  42 STO
390  53  53
391  42 STO
392  54  54
393  42 STO
394  55  55
395  42 STO
396  56  56
397  42 STO
398  59  59
399  42 STO
400  58  58
401  42 STO
402  15  15
403  42 STO
404  14  14
405  42 STO
406  16  16
407  42 STO
408  17  17
409  42 STO
410  18  18
411  42 STO
412  13  13
413  42 STO
414  12  12
415  42 STO
416  57  57
417  42 STO
418  00  00
419  43 RCL
420  19  19
421  22 INV
422  77  GE
423  04  04
424  77  77
425  43 RCL
426  07  07
427  91 R/S
428  61 GTO
429  53  (
430  76 LBL
431  43 RCL
432  43 RCL
433  20  20
434  61 GTO
435  38 SIN
436  76 LBL
437  42 STO
438  43 RCL
439  16  16
440  61 GTO
441  55  ÷
442  76 LBL
443  65  ×
444  43 RCL
445  15  15
446  61 GTO
447  75  -
448  76 LBL
449  85  +
450  43 RCL
451  13  13
452  61 GTO
453  95  =
454  76 LBL
455  60 DEG
456  43 RCL
457  12  12
458  61 GTO
459  95  =
460  76 LBL
461  80 GRD
462  85  +
463  01  1
464  95  =
465  61 GTO
466  70 RAD
467  76 LBL
468  90 LST
469  43 RCL
470  59  59
471  61 GTO
472  25 CLR
473  91 R/S
474  61 GTO
475  53  (
476  00  0
```

Bild 2. Programm-Listing für das Rechner-Roulett

Hans Werner

Mikrocomputer

Eine kleine Einführung

19. Teil

Unmittelbare Adressierung

Struktur:

Operation	Konstante
1. Byte	2. Byte

Für „unmittelbar" wird oft der englische Ausdruck „immediate" benutzt.

Beispiel:

AD k	07 H

Addiere zum Inhalt des Akkumulators die Zahl 07 H; Kurzschreibweise: (AC)+ 07 in AC

Im Adreßteil des Befehls steht eine Konstante und keine Speicheradresse.

Die Konstante kann eine dezimale (D), eine hexadezimale (H) oder eine binäre Zahl sein, z. B.

```
LA  AC  k        10001101  B
LA  AC  k             A1   H
LA  AC  k            100   D
```

Indizierte Adressierung

Struktur:

Operation	Distanzadresse
1. Byte	2. Byte

Beispiel:

AD i	07 H

(Im Indexregister steht 100 D)

Addiere den Inhalt des Akkumulators zum Inhalt der Speicherzelle 107 D. Die Adresse 107 ist die Summe aus:

```
  Inhalt von IR
+ Distanzadresse
=          107
```

Kurzschreibweise: (AC)+ ((IR)+ 07) in AC

Im Adreßteil des Befehls steht die Distanz. Sie wird zum Inhalt des Indexregisters addiert. Die Summe ergibt die wirkliche Adresse.

Indizierte Befehle erlauben mit zwei Bytes den Zugriff zu Speicherplätzen, die um 255 höher liegen können als der Inhalt des Indexregisters. Dadurch werden 3-Byte-Befehle vermieden, d. h. Speicherplätze eingespart. Ein typisches Beispiel hierfür ist der Mikroprozessor 6800 (vgl. Heft 6/1979, Seite 333).

Relative Adressierung

Die relative Adressierung soll nur kurz erwähnt werden. Sie tritt bei Sprungbefehlen auf und ist nicht im Befehlsvorrat unseres Modells enthalten. Im Adreßteil steht die Sprungweite und nicht (wie gewohnt) das Sprungziel.

Besteht z. B. der Adreßteil aus einem Byte, kann die Sprungweite zwischen –127 und + 127 liegen. Ein Bit ist für das Vorzeichen reserviert.

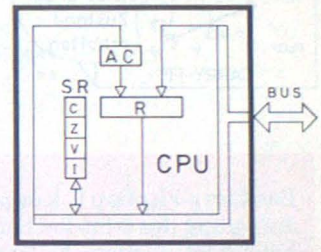
Inhärente oder implizierte Adressierung

Befehle mit implizierter Adressierung enthalten schon im Operationsteil die Adresse; z. B. erfordert der Befehl „Negiere den Inhalt des Akkus" keinen Adreßteil, denn es kann nur der Akku gemeint sein. Solche Begriffe haben mit dem Verständnis für Mikroprozessoren nichts zu tun. Man kann sie getrost vergessen.

Das Status-Register

Rechenergebnisse weisen drei charakteristische Merkmale auf:

1. es entsteht ein Übertrag
   bzw. kein Übertrag
2. das Ergebnis ist Null
   bzw. ist nicht Null
3. Ergebnis ist positiv
   bzw. ist negativ



Diese Angaben sind für die weitere Verarbeitung der Daten äußerst wichtig. Sie werden durch das Rechenwerk ermittelt und in dem Statusregister SR gespeichert (Status = Zustand).

Das Statusregister zeigt den augenblicklichen Zustand der CPU an: Vorzeichen, Übertrag, Ergebnis gleich Null, Interrupt.

Es besteht bei unserem Modell aus 4 Flipflops, die abhängig vom Ergebnis der letzten Rechenoperation gesetzt

oder nicht gesetzt sind. Aufwendigere Mikroprozessoren können sechs und mehr verschiedene Zustände anzeigen.

Übertrag (Carry)

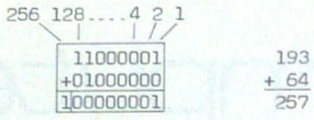
Ist das Ergebnis einer Operation größer als 255, läßt es sich nicht mehr mit 8 bit darstellen. Es entsteht ein Übertrag, und das Status-Bit C wird auf 1 gesetzt (C = Carry = Übertrag). Das kann z. B. bei der Addition von zwei Bytes auftreten.

1. Ergebnis kleiner als 255:

```
 10000001   129
+01000000  + 64
 011000001  193
```

Der Übertrag ist 0, und C wird auch auf 0 gesetzt.

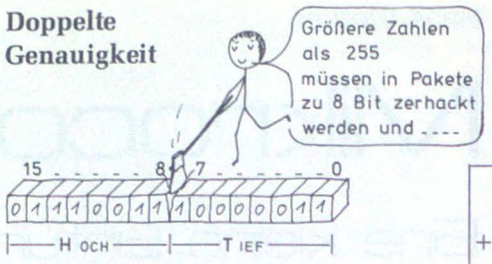
2. Ergebnis größer als 255:



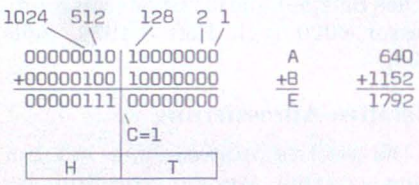
Der Übertrag ist 1, und C wird auch auf 1 gesetzt.

Das Carry-Flipflop C kann als Verlängerung des 8-bit-Rechenwerkes betrachtet werden. Es wird nach jeder Rechnung auf den Wert 0 bzw. 1 des Übertrags gesetzt.

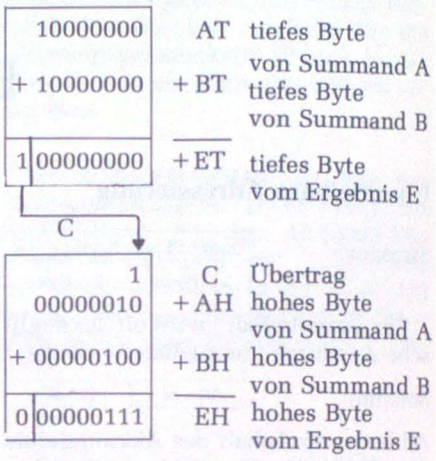
Doppelte Genauigkeit



nacheinander vom Rechenwerk verarbeitet werden. Das ist nur mit Hilfe des Übertrags C möglich. Die Aufgabe 640 + 1152 kann in 2 Teile T und H zerlegt werden. Die Zahlenhälften AT bzw. BT (Byte T von A bzw. B) und AH bzw. BH (Byte H von A, B) werden nacheinander addiert.



Zwischen Bit 7 und 8 tritt ein Übertrag auf, der bei der zweiten Addition zu berücksichtigen ist.



Inhalt des Carry-FF nach der Rechnung

(Fortsetzung folgt)

Ing. (grad.)
Klaus-Dieter Wittek

Memory

Das vorliegende BASIC-Programm ermöglicht es, mit einem PET-2001 „Memory" zu spielen. Bei diesem Spiel liegen 2 x 32 Karten verdeckt auf einem Spielfeld. Abwechselnd können zwei Spieler jeweils zwei Karten aufdecken. Sind die zwei Karten in Farbe und Wert identisch, so wird dem betreffenden Spieler ein Punkt gutgeschrieben, und er kann nochmals zwei Karten aufdecken. Die zwei gleichwertigen Karten werden vom Spielfeld genommen. Sind die aufgedeckten Karten dagegen ungleich, so werden sie wieder verdeckt.

Bedienung des Programms

Mit der Eingabe „RUN" wird das Spiel gestartet, daraufhin erfolgt der Ausdruck:

DIE KARTEN WERDEN GEMISCHT, BITTE WARTEN

In dieser Zeit speichert der Computer mit Hilfe von Zufallszahlen alle 64 Karten in einem Feld A, einer 8 x 8-Matrix, ab. Außerdem wird ein Feld B erzeugt, ebenfalls eine 8 x 8-Matrix, in der alle Matrixelemente durch zwei Grauraster-zeichen belegt werden; dies symbolisiert eine umgedrehte Karte. Ist dieser Vorgang abgelaufen, so druckt der Computer eine kurze Spielanleitung aus und wartet nach Ausdruck „START?" auf die Eingabe ‚J'.

Nach dieser Eingabe wird das Feld B auf dem Bildschirm angezeigt *(Bild 1)*. Am oberen Bildschirmrand stehen dabei die Anzahl der bisherigen Mißerfolge, die Nummer des Spielers, der mit der Wahl an der Reihe ist, und darunter der aktuelle Spielstand.

Nach dem Ausdruck „1. KOORDINATE?" kann mit der Eingabe von zwei Zahlen (Koordinatenwerten) jedes beliebige Feld, d. h. auch jede beliebige Karte, auf dem Spielfeld angesprochen werden. Zunächst muß dabei die Zahl die am linken Spielfeldrand steht, ein-

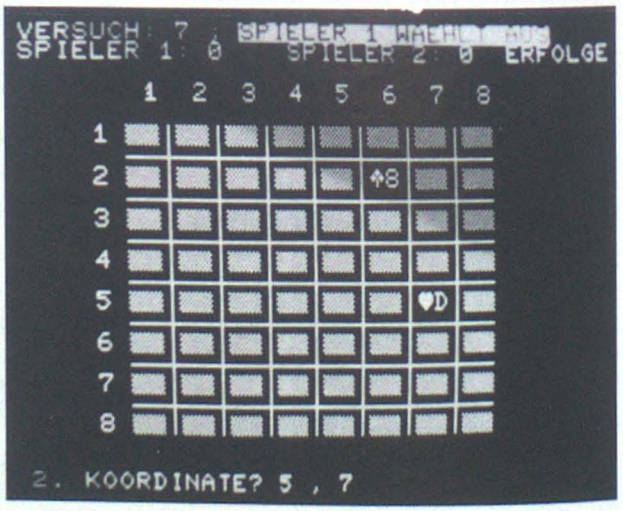


Bild 1. So sieht der PET-Bildschirm während des Memory-Spiels aus (Foto: Knoll)

gegeben werden. Sofort erscheint auf dem Bildschirm ein Feld mit einer aufgedeckten Karte und der Ausdruck „2. KOORDINATE?", der wiederum durch die Eingabe zweier Zahlen beantwortet werden muß. Das danach dargestellte Spielfeld enthält zwei aufgedeckte Karten, und es erscheint der Ausdruck „WEITER?". Die dadurch erzeugte Pause soll dazu dienen, sich die Karten, falls sie ungleich waren, einzuprägen, um sie zu einem späteren Zeitpunkt, wenn die Lage der anderen, gleichwertigen Karten bekannt wird, wiederum aufzudecken.

Wird der Ausdruck „WEITER?" mit ‚J' bedient, so werden die Karten, falls sie ungleich waren, wieder umgedreht, andernfalls gelöscht. Das Spielende ist erreicht, wenn alle Kartenfelder gelöscht sind. Der Computer druckt dann den Endspielstand aus, gratuliert dem Spieler, der gewonnen hat und fragt mit dem Ausdruck „NOCH EIN SPIEL?", ob es weitergehen soll. Mit der Eingabe ‚J' wird ein neues Spiel gestartet, d. h. ein

```
10 REM**MEMORY PROGRAMM(2Y.. 11. 78)**
20 DIMA$(8,8)
30 DIMB$(8,8)
40 DIMC$(8)
50 DIMD$(4)
60 DIME$(64)
70 REM**LADE KARTENWERTE**
80 C$(1)="A"
90 C$(2)="K"
100 C$(3)="D"
110 C$(4)="B"
120 C$(5)="Z"
130 C$(6)="9"
140 C$(7)="8"
150 C$(8)="7"
160 REM**LADE KARTENFARBEN**
170 D$(1)=CHR$(216)
175 PRINT
180 D$(2)=CHR$(193)
190 D$(3)=CHR$(211)
200 D$(4)=CHR$(218)
210 REM**LADE 2 * 32 KATENWERTE**
220 I=1
230 FORJ=1TO8
240 FORK=1TO4
250 E$(I)=D$(K)+C$(J)
260 I=I+1
270 NEXTK
280 NEXTJ
290 I=33
300 FORJ=1TO8
310 FORK=1TO4
320 E$(I)=D$(K)+C$(J)
330 I=I+1
340 NEXTK
350 NEXTJ
360 REM**LADE ANFANGSPARAMETER**
370 RE=0
380 RZ=0
390 P=1
395 PRINT"♥ DIE KARTEN WERDEN GEMISCHT,BITTE WARTEN"
400 REM**LADE MEMORY-FELD A MIT**
410 REM**HILFE VON ZUFALLSZAHLEN**
420 I=1
430 FOR J=1TO8
440 FOR K=1TO8
450 Y=.5
460 Y=RND(Y)
470 YN=Y
480 IFYN=1THENYN=0.9999
490 Z=INT(YN*(65-I))+1
500 A$(J,K)=E$(Z)
510 S=Z
520 H$=E$(Z)
530 IFS>64-ITHEN570
540 E$(S)=E$(S+1)
550 S=S+1
560 GOTO530
570 I=I+1
580 E$(S)=H$
590 NEXTK
600 NEXTJ
610 REM**LADE MEMORYFELD B, DIESES**
620 REM**FELD WIRD AUSGEDRUCKT    **
630 FORJ=1TO8
640 FORK=1TO8
650 B$(J,K)="▒▒"
660 NEXTK
670 NEXTJ
680 REM**SCHREIBE SPIELANLEITUNG**
690 PRINT"♥S"
700 PRINT
710 PRINT
720 PRINT"SPIELANWEISUNG-MEMORY"
730 PRINT"====================="
740 PRINT"DURCH DIE EINGABE ZWEIER KOORDINATEN"
745 PRINT"SOLL VERSUCHT WERDEN, GLEICHARTIGE"
750 PRINT"KARTENWERTE  ZU FINDEN."
760 PRINT"MISSLINGT DIESES, WERDEN DIE KARTEN
    WIE-DER VERDECKT"
770 PRINT"DER ERSTE KOORDINATENWERT STEHT AM
    LINKEN SPIELFELDRAND"
780 PRINT"SPIELEN 2 SPIELER GEGENEINANDER,SO WIRD"
790 PRINT"DEM SPIELER, DER GLEICHARTIGE KARTEN-"
800 PRINT"WERTE FINDET, EIN PUNKT GUTGESCHRIEBEN"
810 PRINT"UND ER KANN EINEN WEITEREN VERSUCH"
815 PRINT"MACHEN."
820 PRINT"DIE FRAGEN:START?, WEITER? UND NOCH EIN SPIEL?"
830 PRINT"MUESSEN MIT DER EINGABE 'J' QUITTIERT"
840 PRINT"WERDEN"
860 REM**START DES SPIELPROGRAMMS"
870 PRINT"START?"
880 GETG$
890 IFG$="J"THEN910
900 GOTO880
910 PRINT"♥"
920 REM**AUSGABE FELDB AM BILDSCHIRM**
930 REM**ALLE KARTEN VERDECKT       **
950 GOSUB2000
960 PRINT"                          ";
970 PRINT"|||||||||||||||||||||||||";
980 REM**EINLESEN DER ERSTEN KARTENWERTE**
990 PRINT"1. KOORDINATE?";
1000 GETXX$
1010 NN=VAL(XX$)
1020 IF NN=0 THEN 1000
1030 IF NN>8 THEN 3000
1040 PRINTNN;
1050 PRINT",";
1060 GETYY$
1070 MM=VAL(YY$)
1080 IF MM=0 THEN 1060
1090 IF MM>8 THEN 3000
1100 PRINTMM
1110 N=NN
1120 M=MM
1130 IFB$(N,M)="▒▒"THEN1150
1140 GOTO3000
1150 B$(N,M)=A$(N,M)
1160 REM**AUSGABE FELD B AM BILDSCHIRM**
1170 REM**1 KART AUFGEDECKT          **
1180 GOSUB2000
1190 PRINT"                          ";
1200 PRINT"|||||||||||||||||||||||||";
```

Bild 2. Das vollständige BASIC-Programm für das Memory-Spiel

anders belegtes Feld A muß erraten werden. Mit der Eingabe ‚N' erscheint der Ausdruck „SPIELENDE" und das Programm ist beendet.

Sämtliche Fehleingaben, wie z. B. das Ansprechen eines gelöschten Feldes o. ä., werden durch den Ausdruck „FEHLER IN DER EINGABE" angezeigt. *Bild 2* enthält das vollständige BASIC-Programm.

Biorhythmus-Berechnung mit dem TI-59

In FUNKSCHAU 1978, Heft 22, Seite 1102, war ein BASIC-Programm zur Biorhythmus-Berechnung abgedruckt. Die hier vorgestellte Version bedient sich des programmierbaren Taschenrechners TI-59, dessen Innereien in *Bild 1* zu sehen sind, und des dazugehörigen Druckers PC-100A/B.

Da das Programm (*Bild 2*) das Software-Modul (Master, ML-20) benutzt, ist die Handhabung denkbar einfach. Berechnet und ausgedruckt wird jeweils ein ganzer Monat, dessen Länge mit B' zu 28, 29, 30 oder 31 Tagen eingegeben wird. Mit A erfolgt die Eingabe des Geburtstages und mit B die des letzten Ta-

```
1210 REM**EINLESEN DER 2. KOORDINATE**
1220 PRINT"2. KOORDINATE?";
1230 GET ZZ$
1240 LL=VAL (ZZ$)
1250 IF LL=0 THEN 1230
1260 IF LL>8 THEN 3000
1270 PRINT LL;
1280 PRINT",";
1290 GETYY$
1300 OO=VAL(YY$)
1310 IF OO=0 THEN 1290
1320 IF OO>8 THEN 3000
1330 PRINTOO
1340 L=LL
1350 O=OO
1360 IF B$(L,O)="■"THEN1380
1370 GOTO3000
1380 B$(L,O)=A$(L,O)
1390 REM**AUSGABE FELD B AM BILDSCHIRM**
1400 REM**2 KARTEN OFFEN                **
1410 GOSUB2000
1420 REM**VERZOEGERUNG ZUM EINPRAEGEN DER**
1430 REM**KARTENWERTE                   **
1440 PRINT"                          ";
1450 PRINT"||||||||||||||||||||||||||";
1460 PRINT"WEITER?"
1470 GETYY$
1475 IF YY$="N" GOTO1780
1480 IFYY$="J"THEN1540
1490 GOTO1470
1500 REM**VERGLEICH DER BEIDEN KARTEN**
1510 REM**WERTE. WAREN SIE GLEICH WER-**
     1520REM**DEN DIE FELDER GELOESCHT    *
1530 REM**ANDERENFALLS WIEDER VERDECKT**
1540 IFA$(N,M)=A$(L,O)THEN1580
1550 B$(N,M)="■"
1560 B$(L,O)="■"
1570 GOTO1690
1580 B$(N,M)="  "
1590 B$(L,O)="  "
1600 IF INT(P/2)<>P/2 THEN1650
1610 REM**SPIELER 2 ERFOLGREICH**
1620 RZ=RZ+1
1630 GOTO1700
1640 REM**SPIELER 1 ERFOLGREICH**
1650 RE=RE+1
1660 GOTO1700
1670 REM**ERHOEHE ANZAHL DER MISSERFOLGE**
1680 REM**SETZE PARAMETER ZURUECK      **1690P=P+1
1690 P=P+1
1700 N=0
1710 M=0
1720 L=0
1730 O=0
1740 REM**ABFRAGE OB 32 ERFOLGE, DAMIT**
1750 REM**WAERE SPIEL BEENDET          **
1760 IFRE+RZ=32THEN4000
1770 GOTO 950
1780 PRINT"NOCH EIN SPIEL?"
1790 GET Q$
1800 IFQ$="J"THEN370
1810 IFQ$="N"THEN1830
1820 GOTO1790
1830 END

2000 REM**SUBROUTINE: AUSGABE DES B-FELDES**
2010 REM**AUF DEM BILDSCHIRM           **
2020 REM**************************************
2030 PRINT"S"
2040 IF INT(P/2)=P/2 THEN 2080
2050 REM**SPIELER 1 BZW. SPIELER 2          **
2060 PRINT;"VERSUCH:"P"; RSPIELER 1 WAEHLT AUS_";
2070 GOTO2090
2080 PRINT;"VERSUCH:"P"; RSPIELER 2 WAEHLT AUS_";
2090 PRINT
2110 REM**SPIELSTANDANZEIGE          **
2120 PRINT"SPIELER 1:";RE;"|    SPIELER 2:";RZ;"|  ERFOLGE"
2140 PRINT
2150 REM**FELD B AUSGABE             **
2160 PRINT"       1  2  3  4  5  6  7  8"
2170 PRINT
2180 FOR I=1 TO 8
2190 PRINT"   ";I;
2200 FOR K=1TO8
2210 IF K<8THEN2240
2220 PRINTB$(I,K)
2230 GOTO2250
2240 PRINTB$(I,K);"|";
2250 NEXTK
2260 IF I<8 THEN 2290
2270 GOTO2300
2290 PRINT"      ——+——+——+——+——+——+——+——"
2300 NEXTI
2310 PRINT
2320 PRINT
2330 RETURN
3000 REM**ANZEIGE EINER FEHLEINGEBE
3020 IFB$(N,M)="  "THEN3060
3030 IFB$(L,O)="  "THEN3050
3040 B$(L,O)="■"
3050 B$(N,M)="■"
3060 PRINT"♥S"
3070 FORW=1TO15
3080 PRINT"S"
3090 PRINT
3100 PRINT
3110 PRINT
3180 IF INT(W/2)=W/2THEN3210
3190 PRINT"    _FEHLER IN DER EINGABE"
3200 GOTO3220
3210 PRINT"    RFEHLER IN DER EINGABE"
3220 NEXT
3230 GOTO 910
4000 PRINT"♥SIEGEREHRUNG--SIEGEREHRUNG--SIEGEREHRUNG"
4010 PRINT"============  ============  ============"
4020 PRINT
4030 PRINT
4040 PRINT"SPIELER 1:";RE;"ERFOLGE"
4050 PRINT"SPIELER 2:";RZ;"ERFOLGE"
4060 PRINT
4065 IF RZ=RE THEN 4100
4070 IF RZ>RE THEN 4090
4080 PRINT"HERZLICHEN GLUECKWUNSCH DEM SPIELER1"
4085 GOTO1780
4090 PRINT"HERZLICHEN GLUECKWUNSCH DEM SPIELER 2"
4100 GOTO1780
READY.
```

000	76	LBL	072	02	02
001	17	B'	073	43	RCL
002	42	STO	074	11	11
003	00	00	075	65	×
004	25	CLR	076	53	(
005	42	STO	077	03	3
006	16	16	078	06	6
007	92	RTN	079	00	0
008	22	INV	080	55	÷
009	58	FIX	081	02	2
010	76	LBL	082	03	3
011	11	A	083	54	)
012	36	PGM	084	95	=
013	20	20	085	38	SIN
014	11	A	086	42	STO
015	92	RTN	087	14	14
016	76	LBL	088	02	2
017	12	B	089	06	6
018	36	PGM	090	69	OP
019	20	20	091	04	04
020	12	B	092	43	RCL
021	76	LBL	093	14	14
022	13	C	094	69	OP
023	36	PGM	095	06	06
024	20	20	096	43	RCL
025	13	C	097	11	11
026	85	+	098	65	×
027	01	1	099	53	(
028	95	=	100	03	3
029	42	STO	101	06	6
030	11	11	102	00	0
031	42	STO	103	55	÷
032	15	15	104	02	2
033	76	LBL	105	08	8
034	15	E	106	54	)
035	01	1	107	95	=
036	44	SUM	108	38	SIN
037	11	11	109	42	STO
038	44	SUM	110	14	14
039	16	16	111	03	3
040	43	RCL	112	06	6
041	11	11	113	69	OP
042	75	-	114	04	04
043	43	RCL	115	43	RCL
044	15	15	116	14	14
045	95	=	117	69	OP
046	42	STO	118	06	06
047	14	14	119	43	RCL
048	22	INV	120	11	11
049	58	FIX	121	65	×
050	03	3	122	53	(
051	07	7	123	03	3
052	01	1	124	06	6
053	03	3	125	00	0
054	02	2	126	55	÷
055	02	2	127	03	3
056	00	0	128	03	3
057	00	0	129	54	)
058	69	OP	130	95	=
059	04	04	131	38	SIN
060	43	RCL	132	42	STO
061	14	14	133	14	14
062	69	OP	134	02	2
063	06	06	135	02	2
064	10	E'	136	69	OP
065	97	DSZ	137	04	04
066	00	00	138	43	RCL
067	15	E	139	14	14
068	91	R/S	140	69	OP
069	76	LBL	141	06	06
070	10	E'	142	98	ADV
071	58	FIX	143	92	RTN

Bild 2. Programm zur Biorhythmus-Berechnung für einen ganzen Monat. Das Ergebnis wird auf dem Drucker PC-100 A ausgegeben

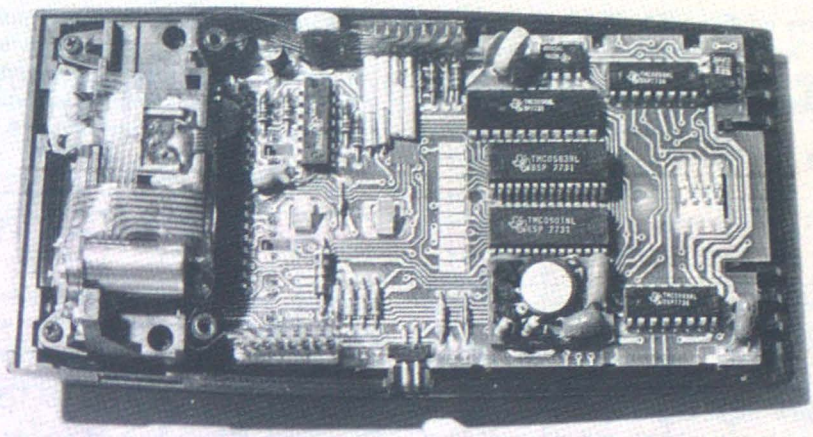


Bild 1. Der Rechner TI-59 von innen

ges vor dem untersuchenden Monat, wobei der Druck auf B gleichzeitig das Programm und den Drucker startet. Der Ausdruck erfolgt in der Form, daß vor jedem Wertesatz eine Leeerzeile erscheint; dann folgen die Nummer des Tages (1...31) und die drei Biorhythmus-Werte.

Nach der Theorie ist der Geburtstag dann mitzuzählen, wenn die Geburt vor 12 Uhr (mittags) erfolgte. Das Programm ist hierfür ausgelegt; das Unterprogramm ML-20 berücksichtigt den ersten Tag nicht. Dieser wird deshalb mit 1 SUM 11 zu Anfang des Programms hinzuaddiert, weil häufig die Geburtsstunde nicht bekannt ist.

Sicher läßt sich das Programm noch verkürzen und optimieren. Gemäß dem Ratschlag im TI-Handbuch wurde aber darauf verzichtet, weil es einwandfrei läuft, einfach zu handhaben ist und die Rechnerkapazität bei weitem noch nicht ausnützt. *Friedrich Wagner*

Jörg Ullrich

Z 80 steuert TI-59

Die hier beschriebene Schaltung wurde entwickelt, um den PC-100A als Drucker für einen Mikroprozessor verwenden zu können. Durch Erweitern der Schaltung kann man aber ohne weiteres alle Funktionen des TI-59 mit dem Mikroprozessor ansteuern.

Die Tastatur des TI-59 besteht aus einer Matrix (*Bild 1*) mit 9 Zeilen und 5 Spalten. Der Rechner schickt der Reihe nach an die Matrix Reihenimpulse (*Bild 2*) mit einer Amplitude von ca. 10 V, die er bei gedrückten Tasten aus den Matrixspalten empfängt.

Je nachdem, welcher Impuls aus welcher Spalte empfangen wurde, weiß der Rechner, welche Funktion ausgeführt werden muß (etwa: Impuls Nr. 2 an Spalte Nr. 3 = ln x). Ersetzt man nun die mechanische durch eine elektronische Tastatur, die dann an die Ausgabestelle eines µC angeschlossen wird, so hat man eine einfache Möglichkeit, den Rechner über ein Programm anzusteuern.

Bild 3 zeigt die elektronischen Schalter, die aus PNP-Transistoren bestehen. Ihre Basen werden über einen Vorwiderstand (zur Strombegrenzung) von einem Binär-Decoder (74154), der die vom µC kommenden Steueranweisungen (*Tabelle 1*) decodiert, angesteuert. Die in der Schaltung enthaltenen Emitterdioden sind notwendig, um die Spalten in dieser Transistormatrix voneinander zu trennen.

Die Anschlußstellen (*Bild 4*) für die Tastatur im TI-59 erkennt man im Batteriefach des Rechners, wobei eine durch das Gehäuse verdeckt ist.

Das in *Bild 5* aufgelistete Programm dient zum Betrieb des PC-100A (TI-59)

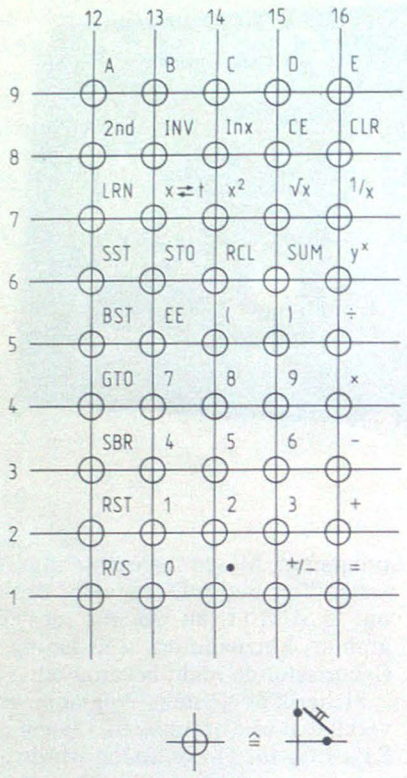


Bild 1. Tastenmatrix des programmierbaren Taschenrechners TI-59

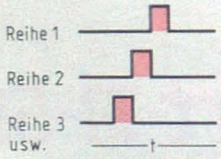
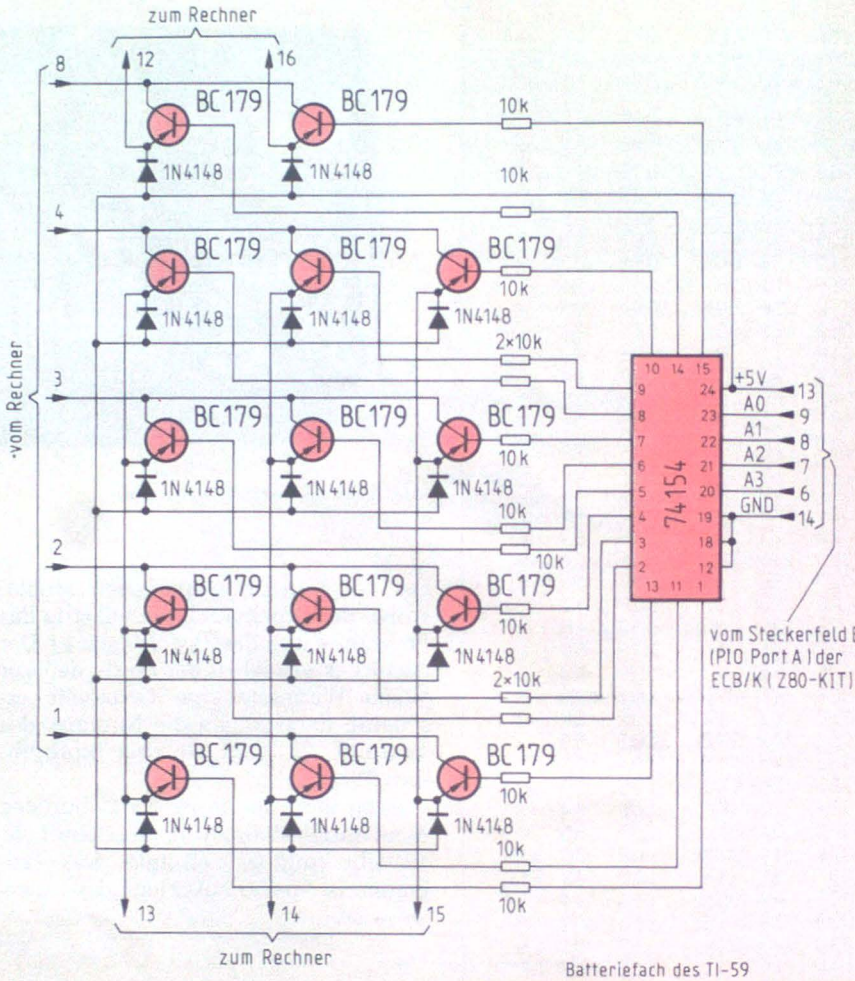


Bild 2. Der Rechner fragt nacheinander die einzelnen Spalten der Tastenmatrix ab



▲ Bild 3. Anschalten des Mikrocomputers an die Rechner-Tastenmatrix

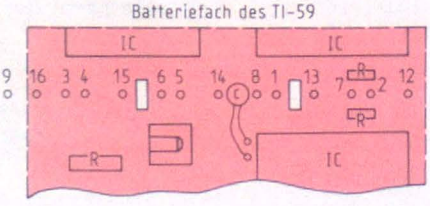


Bild 4. ▶ Lage der Anschlüsse im Batteriefach des TI-59 (die Anschlußnummern beziehen sich auf die in Bild 1 und 3 angeführten Anschlüsse)

```
LOC OBJ CODE    STMT
4010 CB3F        0001
4012 CB3F        0002
4014 CB3F        0003
4016 CB3F        0004
4018 C9          0005
4019 CB27        0006
401B CB27        0007
401D CB27        0008
401F CB27        0009
4021 C9          0010
4022 0611        0011
4024 C5          0012
4025 06FF        0013
4027 10FE        0014
4029 C1          0015
402A 10F8        0016
402C C9          0017
402D 0601        0018
402F C5          0019
4030 CD2240      0020
4033 C1          0021
4034 10F6        0022
4036 C9          0023
4037 CD2240      0024
403A D308        0025
403C 3EFF        0026
403E CD2240      0027
4041 D308        0028
4043 C9          0029
4044 DD210040    0030
4048 0600        0031
404A B8          0032
404B 2805        0033
404D DD23        0034
404F 04          0035
4050 18F8        0036
4052 DD7E00      0037
4055 CD1040      0038
4058 CD3740      0039
405B DD7E00      0040
405E CD1940      0041
4061 CD1040      0042
4064 CD3740      0043
4067 C9          0044
4068 79          0045
4069 CD1040      0046
406C CD4440      0047
406F 79          0048
4070 CD1940      0049
4073 CD1040      0050
4076 CD4440      0051
4079 C9          0052
407A 3E0C        0053
407C CD3740      0054
407F 3E09        0055
4081 CD3740      0056
4084 CDA540      0057
4087 79          0058
4088 CD1040      0059
408B CD3740      0060
408E 79          0061
408F CD1940      0062
4092 CD1040      0063
4095 CD3740      0064
4098 CDA540      0065
409B C9          0066
409C 0E05        0067
409E CD7A40      0068
40A1 CD2D40      0069
40A4 C9          0070
40A5 060F        0071
40A7 C5          0072
40A8 CD2240      0073
40AB C1          0074
40AC 10F9        0075
40AE C9          0076
40AF 0E00        0077
40B1 CD7A40      0078
40B4 CD2D40      0079
40B7 C9          0080
40B8 4C          0081
40B9 CD6840      0082
40BC 4D          0083
40BD CD6840      0084
40C0 3E00        0085
40C2 CD3740      0086
40C5 3E00        0087
40C7 CD3740      0088
40CA 0E01        0089
40CC CD7A40      0090
40CF C9          0091
```

Bild 5. Die Unterprogramme nach Tabelle 2, vom Drucker PC-100 zusammen mit dem Rechner TI-59 aufgelistet. Die Ausgabeadressen beziehen sich auf den Z80-KIT

an dem Z80. Es besteht aus mehreren Unterprogrammen (*Tabelle 2*), durch deren Aufruf die meisten Druckfunktionen des Druckers ausgeführt werden. Die Sprünge, die in den Unterprogrammen enthalten sind, sind relative Sprünge, so daß beim Umschreiben in einen anderen Speicherbereich nur Unterprogrammaufrufe und Adressenladebefehle geändert werden müssen. Vor Anwendung der Unterprogramme, die übrigens durch Anwendung eines entsprechenden Hauptprogramms zu dieser Auflistung benutzt wurden, muß Port A der PIO in den Ausgabezustand versetzt werden (3E, 0F, D3, 0A).

Tabelle 1: Code der wichtigsten Tastenfunktionen

Tasten-Name	entsprechende auszugebende Zahl
0/Dsz	00
1/If flg	01
2/D.MS	02
3/π	03
4/x≧	04
5/Σ +	05
6/x̄	06
7/x = t	07
8/Nop	08
9/Op	09
·/Adv	0A
+/– / Prt	0B
2nd	0C
CLR	0D

Tabelle 2: Unterprogramme

CD1040 Abtrennen von einem halben Byte
CD1940 s. o.
CD2240 Zeitschleife (kurz) für Datenausgabe
CD2D40 Zeitschleife (mittellang) für OP
CD3740 in Reg. A übergebenes halbes Byte (Bit 0, 1, 2, 3) wird ausgegeben
CD4440 notwendig zur Decodierung bei Ausgabe Reg. C (Tabelle 3)
CD6840 im Reg. C übergebenes Byte wird dec. und ausgegeben
CD7A40 vorher in Reg. C übergebene OP-Nr. wird durchgeführt
CD9C40 führt OP 05 durch
CDA540 Zeitschleife (lang) für OP 05
CDAF40 führt OP 00 durch
CDB840 gibt in Reg. HL stehende Adresse aus

Tabelle 3: Daten für das Decodierunterprogramm

Adresse	Daten
4000	01
4001	02
4002	03
4003	04
4004	05
4005	06
4006	07
4007	10
4008	11
4009	12
400A	13
400B	14
400C	15
400D	16
400E	17
400F	21

Datenverschieben mit dem µP 6800

Bei der Entwicklung von Programmen für den Mikrocomputer ist es zweckmäßig, wenn man Daten oder Programme blockweise verschieben kann, z. B. wenn man merkt, daß noch ein Befehl eingeschoben werden muß, wenn ein Befehl überflüssig ist oder wenn man beim Eintippen eines langen Programms kurz vor Schluß merkt, daß man am Anfang ein Byte vergessen hat.

Das vorliegende Programm (*Bild*) führt diesen RAM-Transfer bei Mikrocomputern mit der CPU 6800 aus. Zunächst werden die Variablen folgendermaßen eingegeben:

A031/A032 Anfangsadresse des Ausgangsbereiches

A033/A034 Endadresse des Ausgangsbereiches

A035/A036 Anfangsadresse des Zielbereiches

A037/A038 maximal zulässige Endadresse des Zielbereiches

(A039/A03A werden als Zwischenspeicher benötigt.)

Als maximal zulässige Endadresse des Zielbereiches wird die letzte freie Adresse im RAM-Bereich eingegeben, bevor das Programm eventuell später benötigte Daten überschreiben würde. Nach dem Start bei 0000 rechnet es zunächst aus, ob der umzuspeichernde Bereich in den vorgesehenen Zielbereich paßt. Wenn der Zielbereich zu klein ist, wird das Programm mit dem Software-Interrupt SWI unterbrochen. Wenn der vorgesehene Zielbereich groß genug ist, so wird überprüft, wie die beiden Anfangsadressen zueinander liegen. Für den Fall, daß sich Ausgangs- und Zielbereich überlappen, muß nämlich erst bestimmt werden, von welchem Ende der Transfer beginnen muß; sonst würden womöglich Bytes überschrieben, die noch nicht transferiert sind. Anschließend berechnet das Programm die reale Endadresse des Zielbereiches und überschreibt mit ihr die maximal zulässige Endadresse des Zielbereiches. Je nach Lage der beiden Anfangsadressen zueinander wird dann der Datentransfer von oben oder unten beginnend durchgeführt. Das

```
0000  CE A031  TRANSF  LDX #A031
0003  E6 02            LDA B 2,X
0005  A6 03            LDA A 3,X
0007  A0 01            SUB A 1,X
0009  E2 00            SBC B 0,X
000B  A7 09            STA A 9,X
000D  E7 08            STA B 8,X
000F  E6 06            LDA B 6,X
0011  A6 07            LDA A 7,X
0013  A0 05            SUB A 5,X
0015  E2 04            SBC B 4,X
0017  A7 07            STA A 7,X
0019  E7 06            STA B 6,X
001B  E1 08            CMP B 8,X
001D  22 0A            BHI ZEAB
001F  27 01            BEQ LBTEST
0021  3F               SWI
0022  A1 09    LBTEST  CMP A 9,X
0024  22 03            BHI ZEAB
0026  27 01            BEQ ZEAB
0028  3F               SWI
0029  A6 09    ZEAB    LDA A 9,X
002B  AB 05            ADD A 5,X
002D  A7 07            STA A 7,X
002F  E6 08            LDA B 8,X
0031  E9 04            ADC B 4,X
0033  E7 06            STA B 6,X
0035  A6 00    VOHIB   LDA A 0,X
0037  A1 04            CMP A 4,X
0039  22 21            BHI TRAVO
003B  26 06            BNE TRAHI
003D  A6 01            LDA A 1,X
003F  A1 05            CMP A 5,X
0041  22 19            BHI TRAVO
0043  FE A033  TRAHI   LDX $A033
0046  A6 00            LDA A 0,X
0048  09               DEX
0049  FF A033           STX $A033
004C  FE A037           LDX $A037
004F  A7 00            STA A 0,X
0051  BC A035           CPX $A035
0054  27 1F            BEQ ENDE
0056  09               DEX
0057  FF A037           STX $A037
005A  20 E7            BRA TRAHI
005C  FE A031  TRAVO   LDX $A031
005F  A6 00            LDA A 0,X
0061  08               INX
0062  FF A031           STX $A031
0065  FE A035           LDX $A035
0068  A7 00            STA A 0,X
006A  BC A037           CPX $A037
006D  27 06            BEQ ENDE
006F  08               INX
0070  FF A035           STX $A035
0073  20 E7            BRA TRAVO
0075  7E E08D  ENDE     JMP RESTAR
```

Programm zum Verschieben von Datenblöcken mit dem Mikroprozessor 6800

Programm endet mit einem Sprung in das Restart-Programm des Betriebsprogramms.

Ohne Änderung kann das Programm auf beliebigen Adressen betrieben werden. Es wurde für den MEK 6800 D2 entwickelt; für Verwendung auf anderen 6800-Systemen müssen die beiden letzten Bytes (E0 8D) durch den Restart-Vektor des jeweiligen Systems ersetzt werden (kann auf den Adressen FFFE/FFFF kontrolliert werden).

Gerhard Latzin

8080 berechnet Primzahlen

Dieses Programm zur Primzahlen-Berechnung wurde für den ITT-Mikroprozessor-Experimenter mit dem 8080A-Chip geschrieben und berechnet fortlaufend alle Primzahlen bis 10000 (dez.).

Primzahlen sind nur durch 1 oder sich selbst teilbar. Das Programm *(Bild 1)* beginnt zunächst mit der Anzeige der Zahl 2; das ist die kleinste und einzige gerade Zahl in der Primzahlenfolge. Danach erfolgt die Untersuchung der Zahl 3. Die zu prüfende Zahl steht im Registerpaar HL. Zunächst wird diese Zahl in den Akkumulator transportiert und um eine Stelle nach rechts verschoben. Dies kommt im Hexadezimalsystem einer Division durch 2 gleich. Da alle zu prüfenden Zahlen im Registerpaar HL ungerade Zahlen sind, wird bei der Division die rechte Stelle aus dem Register geschoben, d. h. der erste Divisor besteht also aus (HL) : 2 – 0,5. Steht beispielsweise die Zahl 11 im Registerpaar HL, so ist der erste Divisor 5. Dieser wird im Registerpaar DE abgelegt. Es erfolgt nun die Division (HL): (DE).

Blieb bei der Division ein Rest, dekrementiert man den Divisor in DE, und es erfolgt eine erneute Division. Dieses Spiel wird so lange fortgesetzt, bis der Divisor 1 geworden ist. War der Dividend in HL durch keinen Divisor ohne Rest teilbar, so hat man eine Primzahl. War die geprüfte Zahl jedoch ohne Rest teilbar, war sie keine Primzahl, und es erfolgt die Untersuchung der nächsten ungeraden Zahl. Hierzu wird HL zweimal inkrementiert, und der Prozeß beginnt von neuem.

Zum Schluß wird die im Hex-System errechnete Primzahl noch in den BCD-Code umgerechnet und zur Anzeige gebracht. Dies geschieht nach jeder Betätigung der „RUN"-Taste, aufsteigend ab der Zahl 2. *Bild 2* zeigt die 8080-Programmliste.

Arno Ruff

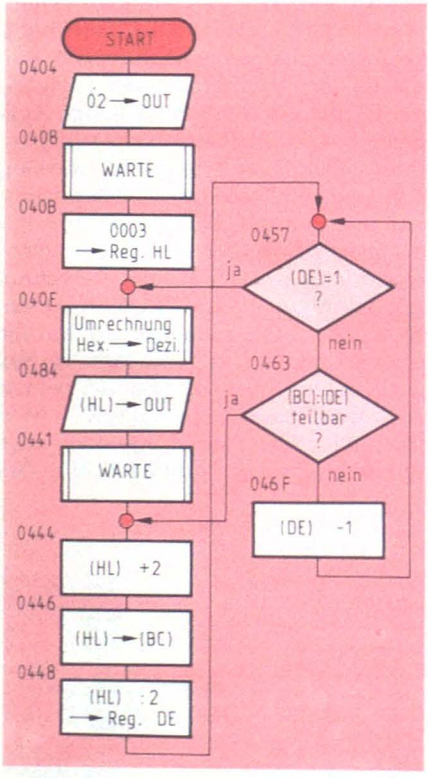


Bild 1. Flußdiagramm zur Primzahlen-Berechnung

```
0400  3E00    MVI A 00
0402  D302    OUT LLAMPE
0404  3E02    MVI A 02
0406  D301    OUT RLAMPE
0408  CD9804  CALL 0498
040B  210300  LXI H 0003
040E  7D      MOV A L
040F  E5      PUSH H
0410  E60F    ANI 0F
0412  47      MOV B A
0413  110000  LXI D 0000
0416  210100  LXI H 0001
0419  CD7F04  CALL 047F
041C  E1      POP H
041D  7D      MOV A L
041E  E5      PUSH H
041F  CD9004  CALL 0490
0422  211600  LXI H 0016
0425  CD7F04  CALL 047F
0428  E1      POP H
0429  7C      MOV A H
042A  E5      PUSH H
042B  E60F    ANI 0F
042D  47      MOV B A
042E  215602  LXI H 0256
0431  CD7F04  CALL 047F
0434  E1      POP H
0435  7C      MOV A H
0436  E5      PUSH H
0437  CD9004  CALL 0490
043A  219640  LXI H 4096
043D  CD7F04  CALL 047F
0440  E1      POP H
0441  CD9804  CALL 0498
0444  23      INX H
0445  23      INX H
0446  44      MOV B H
0447  4D      MOV C L
0448  78      MOV A B
0449  B7      ORA A
044A  1F      RAR
044B  57      MOV D A
044C  79      MOV A C
044D  DA5404  JC 0454
0450  1F      RAR
0451  C35704  JMP 0457
0454  1F      RAR
0455  F680    ORI 80
0457  5F      MOV E A
0458  FE01    CPI 01
045A  C26304  JNZ 0463
045D  7A      MOV A D
045E  FE00    CPI 00
0460  CA0E04  JZ 040E
0463  79      MOV A C
0464  93      SUB E
0465  4F      MOV C A
0466  CA7604  JZ 0476
0469  78      MOV A B
046A  9A      SBB D
046B  47      MOV B A
046C  D26304  JNC 0463
046F  1B      DCX D
0470  44      MOV B H
0471  4D      MOV C L
0472  7B      MOV A E
0473  C35804  JMP 0458
0476  78      MOV A B
0477  92      SUB D
0478  47      MOV B A
0479  CA4404  JZ 0444
047C  C36C04  JMP 046C
047F  05      DCR B
0480  F8      RM
0481  7B      MOV A E
0482  85      ADD L
0483  27      DAA
0484  5F      MOV E A
0485  D301    OUT RLAMPE
0487  7A      MOV A D
0488  8C      ADC H
0489  27      DAA
048A  57      MOV D A
048B  D302    OUT LLAMPE
048D  C37F04  JMP 047F
0490  E6F0    ANI F0
0492  0F      RRC
0493  0F      RRC
0494  0F      RRC
0495  0F      RRC
0496  47      MOV B A
0497  C9      RET
0498  DB04    IN CSHALT
049A  E608    ANI 08
049C  C29804  JNZ 0498
049F  DB04    IN CSHALT
04A1  E608    ANI 08
04A3  CA9F04  JZ 049F
04A6  C9      RET
```

Bild 2. Programmliste für den Mikroprozessor 8080

Hans Werner

Mikrocomputer

Eine kleine Einführung

20. Teil

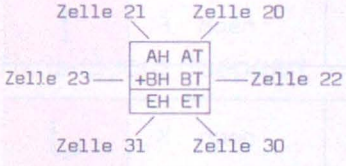
Additionsprogramm

Mit einem kurzen und einfachen Programm kann die doppelte Genauigkeit erzielt werden. Um den Übertrag C addieren zu können, fehlt noch der Transportbefehl:

Carry in den Akku laden	CA AC

Der Befehl ist nur 1 Byte lang und besitzt keine Adresse (inhärenter Befehl). Eines der Merkmale d, k, i ist deshalb nicht erforderlich.

Speicherplatzbelegung:



0	LA	AC	d	20
1	AD		d	22
2	SP	AC	d	30
3	CA	AC		
4	AD		d	21
5	AD		d	23
6	SP	AC	d	31
7	END			

AT in den Akku laden
AT + BT = ET in den Akku
ET in Zelle 30 speichern
Übertrag C in den Akku laden
C + AH
C + AH + BH = EH
EH in Zelle 31 speichern
Stop bei Schritt 7

Nach jeder Addition wird das Carry-Flipflop entweder auf 0 oder 1 gesetzt.

Das Vorzeichen

Werden negative Zahlen als 2^n-Komplement definiert, stellt die höchste Stelle (Bit 7) das Vorzeichen dar. Das 2^n-Komplement ist bei 8 bit die Ergänzung zu $2^n = 256$. Also: Zunächst alle Stellen der positiven Zahl invertieren und anschließend 1 addieren.

```
Bit 7 = 0: pos. Zahl
Bit 7 = 1: neg. Zahl
```

Beispiel: 0 0000010 = +2
1 1111110 = -2

Der Mikrocomputer kann positive und negative Zahlen nicht unterscheiden. Es ist die Aufgabe des Programmierers, das Programm entsprechend zu entwerfen. Als Hilfe wird Bit 7 zusätzlich im Statusregister gespeichert.

Nach allen Rechenoperationen wird das Vorzeichen-Flipflop V auf den Wert der Akkumulatorstelle 7 gesetzt. Der Inhalt von V kann als Vorzeichen interpretiert werden:
0 = positiv, 1 = negativ

Nullmeldung Z

Das Null-Flipflop Z wird gesetzt, wenn das Ergebnis einer Rechenoperation Null ist.
Ergebnis gleich 0: Z=1
Ergebnis ungleich 0: Z=0
(Z = Zero = Null)

Beispiel:

```
Akkuinhalt:    01111010   Z=0
           +   10000001
Akkuinhalt:    11111011   Z=0
           UND 00000100
Akkuinhalt:    00000000   Z=1
```

Die Tabelle zeigt die Eigenschaften verschiedener Halbleiterspeicher. Wie das so in einem Test ist, fehlen die Namen der Speicher. Versuchen Sie bitte, die Kurzbezeichnungen (ROM usw.) nachzutragen.

Kurzname	programmierbar	löschbar	Stromausfall	Einsatz	Stückzahl
	nein	nein	Daten gehen nicht verloren	Programm-speicher	ab 200
	ja	nein			ab 1
	ja	ja			
	Ein-Auslesespeicher		gehen verloren	Daten-speicher	

nach L

A

Sehr gut, die Lösung ist richtig.

nach C

B

Worin unterscheiden sich die beiden Befehle?
Lade den Inhalt von Zelle n in den Akku
Hole den Inhalt des Eingangsregisters in den Akku

☐ a) im Operationsteil
☐ b) im Adreßteil

Bei Lösung
a) nach N
b) nach O

C

Leider nicht. Sie verwechseln wahrscheinlich die unipolare Technik mit der bipolaren. Unipolare Speicher sind z. B. in CMOS-Technik aufgebaut. Wiederholen Sie bitte die Aufgabe.

nach M

D

Das ist richtig. Sehr gut.

nach F

E

123	LA AC d	
124	01	H
125	18	H

Die Tabelle zeigt die Speicherplatzbelegung des Befehls LA AC d 01 18. Die Adresse ist hexadezimal angegeben.

Wird mit den Impulsen AT und AH

☐ a) die Speicheradresse 123 usw. aus der CPU gegeben?

oder

☐ b) die Befehlsadresse 01, 18 in das Datenregister transportiert?

Bei Lösung
a) nach R
b) nach Q

F

Die Antwort ist falsch, denn beide Befehle sind länger als ein Byte. Sie sollten die Aufgabe neu überdenken.

nach P

G

Schade, die Antwort ist zur Hälfte richtig. Überlegen Sie noch einmal gründlich.

nach M

H

Das war bestimmt ein Versehen. Der Befehl AD k 25 ist nur 2 Byte lang. Also steht die 25 in Zelle 9.

nach F

I

Der Befehl SP AC d 200D kann nicht ausgeführt werden, weil die Adresse 200 eine Zelle im ROM anspricht. In diesen kann kein Wert eingespeichert werden.

nach K

J

Jetzt mal was Negatives.
Verwandeln Sie die Binärzahl 00000111 (+7) in –7 (2^8-Komplement).

nach T

K

Von oben nach unten ist die richtige Reihenfolge: ROM, PROM, EPROM, RAM.

nach M

L

Welche Angaben sind für unipolare Speicher zutreffend?
1. Am Strom ist nur eine Ladungsträgerart beteiligt.
2. Die Packungsdichte ist geringer als bei der anderen Technik (weniger Bit/mm²).
3. Relativ langsam
4. Relativ großer Stromverbrauch

☐ a) 1 und 3 treffen zu
☐ b) 2 und 4 treffen zu
☐ c) 2 und 3 treffen zu

Bei Lösung
a) nach B
b) nach D
c) nach H

M

Sie haben unrecht, die Befehle unterscheiden sich im Adreßteil.
LA AC d n: Lade den Inhalt der Zelle n in den Akku
LA AC d 512D: Lade den Inhalt von ER in den Akku

nach P

N

Sie haben richtig gewählt. Die Belohnung ist – wie gemein! – die nächste Aufgabe.

nach P

O

5 LA AC d OTTO AD k 25 In Speicherzelle 5 steht der Befehlsteil LA AC d. In welcher Speicherzelle steht die Zahl 25?	☐ a) in Zelle 6 ☐ b) in Zelle 9 ☐ c) in Zelle 10	Bei Lösung a) nach G b) nach E c) nach I	P
Das stimmt leider nicht. Beide Fälle können leicht miteinander verwechselt werden. Synchron zu den Impulsen AT, AH gibt die CPU die tiefe und hohe Adreßhälfte heraus. Beide werden in den Adreßregistern gespeichert. A7...A0 dienen zur Speicheradressierung. Von den anderen 8 bit werden nur die Stellen A9, A8 benutzt. Mit ihnen wird selektiert. Hoher und tiefer Adreßteil eines Befehls haben mit den Impulsen AH und AT keine direkte Beziehung. Beide Teile werden zum Zeitpunkt L/S aus dem Speicher gelesen.		nach S	Q
Sie haben die richtige Antwort gefunden. Wenn Sie dennoch unsicher sind, können Sie die Erklärung unter Punkt Q nachlesen.		nach S	R
Der Befehl SP AC d 200D kann nicht ausgeführt werden, weil...		nach J	S
Lösung: +7 0 0000111 +7 1 1111000 + 1 −7 1 1111001		–7 ist die Ergänzung von +7 zu $2^8 = 256$, also 249. Die Umwandlung (Konvertierung) kann auch schematisch erfolgen: Negation und +1. Bit 8 gibt das Vorzeichen an.	(Fortsetzung folgt)	T

Hans-Georg Joepgen

Der Euro-Apple

Neben dem PET 2001 und dem Rechner TRS-80 zählt in den Vereinigten Staaten seit Monaten der Hobby-Computer Apple II zu den meistverbreiteten Basic-Maschinen. Einem Einsatz dieses Rechners in Europa stand bisher die Tatsache entgegen, daß der Original-Apple ein Farbfernsehsignal amerikanischer Norm ausgibt. Die Firma ITT Lehr- und Videosysteme in Pforzheim brachte nun, gemeinsam mit weiteren europäischen ITT-Gesellschaften, unter dem Namen „ITT 2020" eine Europa-Version des Apple II auf den Markt.

Der Rechner präsentiert sich als graues Kunststoffgehäuse in Form einer etwas zu groß geratenen Reiseschreibmaschine ohne Walze und Wagen, Abmessungen rund 46 cm x 39 cm mit knapp 12 cm Höhe (*Bild 1*). Eine erweiterte Schreibmaschinentastatur mit normgerechten Tasten dient der Eingabe von Befehlen, Daten und Programmen. Zur Verfügung stehen Stekker zum Anschluß eines Tonbandgerätes oder Kassettenrecorders, weiter ein FBAS-Signal nach Pal-Norm für einen Farbmonitor sowie das Ausgangssignal eines Drahtfunksenders, der einen Farbfernsehempfänger über dessen Antennenbuchse speisen kann.

Oszillator und Modulator belegen als sorgfältig abgeschirmte Baugruppe einen von acht Kartensteckplätzen im Gerät, Plätze, die zur Aufnahme beliebiger Interface-Schaltungen vorbereitet sind und die von der Betriebssoftware bequem angesprochen werden können. Die Stromversorgung der Zusatzbaugruppen und – mit Ausnahme von Tonträgergerät und Bildschirm-Ausgabe – auch die Versorgung von Peripheriegeräten wie Floppy-Disk-Laufwerken übernimmt der Euro-Apple selbst: Er verfügt über ein leistungsfähiges 65-W-Schaltnetzteil mit geringen thermischen Verlusten, das an den Steckplätzen die vergleichsweise hoch belastbaren Spannungen –12 V, –5 V, 5 V und 12 V zur Verfügung stellt.

Über diese Steckplätze hinaus gibt es im Euro-Apple einen bescheiden „Game Connector" – „Spiel-Verbindung" – genannten DIL-Stecker, der Zugang zu insgesamt sieben Low-Power-Schottky-TTL-Eingängen und -Ausgängen ermöglicht, Ports, die der Rechner auf Wunsch mit Daten belegt oder auch abfragt. Weiter an den „Spielstecker-Kontakten": Die Eingänge jener vier Analog-Digital-Wandler, die der Tausendsassa Euro-Apple bereits fest eingebaut enthält. Den Widerstandswert, den der Rechner an diesen Eingängen „sieht" (0...100 kΩ),

Bild 1. Der Mikrocomputer ITT-2020 ist die europäische Pal-Version des Apple-II

setzt er permanent in jeweils ein 8-bit-Datenwort um, das vom Programm oder unmittelbar durch Tastatur-Befehl jederzeit abgeholt, weiterverarbeitet und angezeigt werden kann: ideal zum Anschluß zweidimensionaler Steuerknüppel-Potentiometer.

Firmware und Betriebssystem

Schon an den Eingabe-Ausgabe-Möglichkeiten wird sichtbar, daß der Euro-Apple seinem Programmierer und Operator weitaus mehr Möglichkeiten bietet als die – dafür aber auch deutlich billigeren – Konkurrenten PET und TRS-80. Noch deutlicher wird dieser Unterschied bei der Betrachtung von Firmware und Betriebssystem. Es würde den Rahmen dieses Berichts bei weitem sprengen, wenn auch nur die bedeutenderen der zahlreichen Euro-Apple-Vorzüge auf diesem Gebiet angesprochen würden. Deshalb sei hier die Beschränkung auf jene Eigenschaften erlaubt, die dem Verfasser bei seiner Erprobung des Rechners am meisten Vergnügen bereiteten. Da ist erst einmal die Apple-Fähigkeit, in zweierlei Betriebsarten Rechenergebnisse als farbige Grafiken auszugeben, einmal mit der hohen Auflösung von 69 120 Bildpunkten (kein Druckfehler: über 69 000 Bildpunkte) und zum zweiten in einer weniger speicherplatzkostenden Betriebsart mit der Auflösung 1600 Bildpunkte, von denen ein jeder 16 verschiedene Farben annehmen kann – Schwarz mitgezählt.

Höchst handliche Kommandos erlauben die Manipulation ganzer Bildpunkt-Gruppen, das Schreiben von Kurven und „Ausmalen" von Teilflächen. Ist die Struktur einer grafischen Darstellung erst einmal als Tabelle gespeichert, wobei ein einziges Byte den Zustand von bis zu drei Bildpunkten trickreich und speicherplatzsparend beschreibt, kann man die entsprechende Darstellung in beliebiger Größe an einer frei wählbaren Schirmstelle entstehen lassen – sogar Rotation des Gebildes um einen beliebigen Punkt ist möglich. Dazu noch ein fest eingebauter Lautsprecher, der programmgesteuert klickt, brummt, singt und pfeift: Das ist fürwahr Ausgabe-Komfort!

Fest eingebaut: 6502-Mini-Assembler und Integer-Basic

Nach dem Einschalten präsentiert sich der Euro-Apple als dreisprachig: Er ist bereit, sowohl Kommandos in Maschinensprache als auch in Assembler-Sprache (CPU: 6502) und in einem komfortablen Ganzzahl-Basic-Dialekt entgegenzunehmen. Bleiben wir erst einmal auf der Maschinensprachen-Ebene: Hier bietet das Euro-Apple-Betriebssystem die Möglichkeit, Speicher- und Registerinhalte anzuzeigen und zu verändern, Maschinenprogramme auf Band zu schreiben und zurück zu lesen, diese Programme zu disassemblieren und – wenn gewünscht – auch im Einzelschritt-Modus abzufahren. Sodann der Mini-Assembler: Er erlaubt – mit Ausnahme der symbolischen Adressierung – im wesentlichen alle jene Praktiken, die man sonst nur als zusätzliche (und teure) Software bekommt.

Sowohl vom Monitor als auch vom Assembler aus läßt sich bequem die eingebaute Basic-Version aufrufen, die neben den Standard-Möglichkeiten der Ganzzahl-Basic-Dialekte automatische Zeilennumerierung, die zur Fehlersuche höchst nützliche Betriebsart TRACE (Ausdrucken jeder durchlaufenen Zeilennummer) und einen DSP genannten Befehl bietet. Dieser Befehl, gefolgt von dem Namen einer Variablen, veranlaßt – wann immer im Programm der Wert der Variablen verändert wird – die Anzeige von Zeilennummer und Variablen-Neuwert. Jeder, der schon mal das zweifelhafte Vergnügen einer stundenlangen Fehlersuche in einem Basic-Programm genoß, wird DSP zu schätzen wissen. Sehr angenehm weiterhin: Variablen-Namen dürfen beliebig viele Zeichen umfassen. Dies erleichtert das Schreiben selbstdokumentierender Programme ungemein.

Hochsprachen auf Abruf

An ihre Grenzen stößt die residente Basic-Version im Apple dort, wo Aufgaben höherer mathematischer Komplexität zu lösen sind. Kein Problem: Zum Lieferumfang des ITT-2020 gehört eine Kassette mit einem Fließkomma-

```
666  PRINT " ";: GET A$: HOME : PRINT
     " + PLOT-AENDERUNG";: HTAB 3
     0: PRINT "(P)": PRINT " + KO
     MMANDO-FELD";: HTAB 30: PRINT
     "(K)": PRINT " + RECHNUNGSFO
     RTSETZUNG";: HTAB 30: PRINT
     "(R)": PRINT " + TABELLE (
     T) ODER ENDE";: HTAB 30: PRINT
     "(E) ? ";
667  GOSUB 820: GET A$: IF A$ = "
     P" THEN 675
```

Bild 2. Der Euro-Apple erlaubt, bis zu neun Bildschirmzeilen in eine Basic-Zeile zu packen. Das bedeutet weniger Speicherbedarf und schnellere Programme

Interpreter. Dieser schnell geladene Dialekt, Name „Applesoft", stellt im wesentlichen eine Erweiterung der von Microsoft geschriebenen und auch im PET verwendeten Basic-Abart dar. Nicht-residente Hochsprachen wie Applesoft haben den Nachteil, daß sie einige KByte im RAM belegen, dafür aber auch den Vorteil, daß sie beliebig austauschbar sind. So vertreibt die Firma „The 6502 Program Exchange" in Reno, USA, zum Beispiel (einschließlich eines über hundertseitigen Einführungslehrgangs) für ganze 43 Dollar eine Tonbandkassette mit einem Interpreter der höheren Programmiersprache FOCAL von Digital Equipment, die ein wesentlich komprimierteres und trickreicheres Programmieren erlaubt als Basic. Amerikanische Fachzeitschriften wollen wissen, daß in Kürze weitere höhere Programmiersprachen für Apple und den Euro-Apple ITT 2020 auf den Markt kommen.

Erfahrungen mit dem Euro-Apple

Erfahrungsberichte sind immer eine recht subjektive Angelegenheit. Ihre Aussage hängt ab von den Erwartungen des Autors bei der Erprobung, aber auch von seinen bisherigen Erfahrungen mit ähnlichen Produkten. Deswegen hier ein paar Worte über die Ausgangslage: Der Verfasser blickt auf ein knappes Jahr Arbeit mit seinem (heißgeliebten) PET zurück und kennt beim Entstehen dieses Beitrags vom Rechner TRS-80 allein die Handbücher. Der folgende Bericht wird nach zehnwöchiger Arbeit mit dem Apple erstattet.

Negatives...

Das Gerät, das zur Verfügung stand, war ein Prototyp, ausgestattet mit 32 KByte Schreib-Lese-Speicher, einem ITT Studio-Recorder 65 und einer Mini-Floppy-Station „Disk II", Hersteller Shugart und Apple, Vertrieb ITT Lehr- und Videosysteme Pforzheim. Es fiel eine extrem hohe Empfindlichkeit dieses Prototyps gegen statische Entladungen auf: Beim Berühren wurde öfter ein Zwangs-Reset ausgelöst, der in einigen Fällen zum Programm- und Datenverlust führte. Nach Rücksprache mit ITT Pforzheim wurde eine Änderung der Verbindung Gerätemasse-Netzschutzleiter vorgenommen, was zu deutlicher Besserung führte. Nach Auskunft von ITT Pforzheim ist der Fehler bei Seriengeräten inzwischen beseitigt.

Ärgerlich für jemanden, der den ausgezeichneten PET-Editor gewohnt ist, die umständliche Art, mit der beim 2020 Programme redigiert werden müssen. Das Einfügen eines einzigen vergessenen Zeichens in eine Basic-Zeile, beim PET hochkomfortabel (Cursor positionieren, „Insert"-Taste, vergessenes Zeichen einfügen, Return und fertig), das wird beim Apple zur größeren Operation, die – wenn dazu noch in Textstrings zwischen Anführungszeichen verbessert werden muß – allerlei Tücken zeigt; Tücken, denen man nur mit Eingriffen in das Betriebssystem (Formatierung der Schirmausgabe ändern) beikommt. Mit dem FOCAL-Interpreter im Bauch, der einen vorzüglichen Editor enthält, benimmt sich der Apple jedoch in dieser Beziehung wieder einwandfrei.

Dritter Mangel am ITT 2020: Die gleichzeitige Anwendung von Applesoft und der Grafik-Betriebsart „Hohe Auflösung" war nur mit Einschränkungen möglich, weil der vom Band geladene Interpreter auch solche Speicherbereiche belegt, die beim „Feinstrich-Plotten" vom Betriebssystem beansprucht werden. In den Vereinigten Staaten wird von Apple Computer Incorporated bereits eine residente Applesoft-Version angeboten, die als Einbaukarte Speicher für den Eigenbedarf gleich selbst mitbringt. Wie man hört, sind ITT-Ingenieure zur Zeit damit beschäftigt, eine für Europa bestimmte Pal-Applesoft-Karte zu basteln. Unterdessen dienen zwei als Bandprogramm mitgelieferte und bei Bedarf zu ladende Farbgrafik-Programme als Behelf. Der Verfasser beobachtete mit diesen Routinen gelegentliche Farbart-Abweichungen in Abhängigkeit von der Position des Plot-Punktes. Ob das Eigenheiten des Rechners oder des Farbempfängers waren, wurde nicht weiter untersucht, da der Effekt nur wenig störte.

Nicht sehr vorbildlich benahm sich unser Euro-Apple, was hochfrequente Ausstrahlungen angeht. Mit einem Behelfs-Meßempfänger DR 28 war zwischen 150 kHz und 104 MHz bei 2 m Abstand zwischen Rechner und Empfänger ein ganzer Lattenzaun von Trägern zu hören – da zeigte sich beispielsweise der PET in seinem Stahlblech-Gehäuse wesentlich dezenter.

Als letzter Mangel, der dem „petgewöhnten" Autor auffiel, sei die Tatsache genannt, daß ein Deponieren von Daten und Texten auf Band nur nach dem Nachladen spezieller Maschinenprogramme möglich ist – aber wenn wir schon vom Tonband-Interface des Euro-Apple sprechen: Da fällt der Übergang zu den zahlreichen Vorzügen dieser Maschine nicht schwer.

... und Positives

Der Rechner ITT 2020 ist erst einmal um vieles fixer als der PET, was das Laden von Programmen von Band angeht: 44 Sekunden brauchte ein Testprogramm in Applesoft, das als Commodore-Basic-Version vom PET 2 Minuten 11 Sekunden Ladezeit verlangte. Die PET-Väter ließen dazu verlauten, das läge an der hohen Betriebssicherheit beim PET-Kassetten-Betrieb: Nun, der Euro-Apple zeigte nie einen Ladefehler.

Vorzüglich sind auch die Dokumentationsmaterialien, die dem 2020 bei

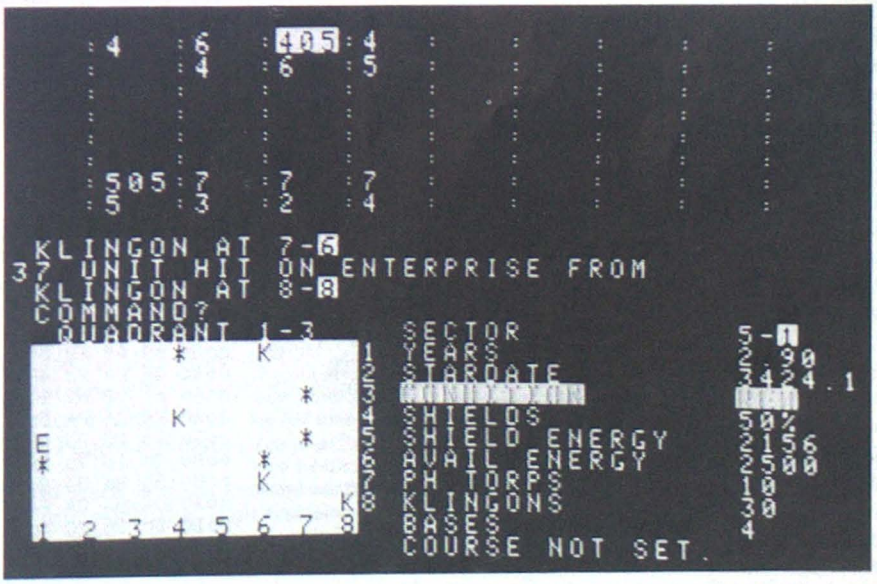


Bild 3. „Enterprise" in kritischer Lage: Die bisher wohl vertrackteste Version des Computerspiels „Startrek" liegt dem ITT-2020 als Kassettenprogramm bei; aus dem Lautsprecher erschallt Schlachtenlärm

der Lieferung beilagen: Ein wunderschön (in Englisch) gemachter Basic-Lehrgang für Anfänger, deutlich besser noch als das zu Recht oft gerühmte TRS-80-Begleitmaterial, ein ausführliches Applesoft-Handbuch dazu, sodann eine 150seitige Materialsammlung mit Schaltbildern, Bauteil-Stücklisten, ausführlicher Besprechung aller Ports, mit einer Bedienungsanweisung für Mini-Assembler und Monitor plus Betriebssoftware-Listing, mit Kurzaufsätzen verschiedener in den Staaten bekannter Hobby-Programmierer, die über spezielle Tricks berichten, mit denen sie ihren Apple zu Sonder- und Höchst-Leistungen anspornten. Dazu, um das Maß voll zu machen, noch gesonderte Betriebsanleitungen und Anwendungshinweise für die mitgelieferten Demonstrations- und Maschinenroutinen-Programme, für das Disk Operating System und das Drucker-Interface nochmal ein eigenes Vorab-Handbuch: PET-Käufer werden so was mit Wehmut hören! Nun zum Rechner selbst:

Beim Programmieren in Applesoft erweist sich als angenehm, daß ohne weiteres speichersparend Zeilen mit bis zu 239 Zeichen Länge eingesetzt werden können (*Bild 2*), andere Rechner streiken bereits bei einem Drittel dieses Wertes.

Die Hauptvorzüge des Euro-Apple liegen jedoch in der hohen Flexibilität seiner Ein- und Ausgabe-Möglichkeiten. Das Steckplatz-Konzept, die Farbgrafik und der Lautsprecher wurden bereits erwähnt, nicht jedoch, was man mit dem Lautsprecher alles machen kann: So erprobte der Autor zum Beispiel ein „Appletalker" genanntes Sprachausgabe-Programm von Mad Hatter Software, Dracut, Massachusetts, das über den Tonband-Anschluß eingegebene Geräusche und menschliche Worte (recht speicherintensiv allerdings) abspeichert und per Aufruf durch Benutzerprogramm über den Lautsprecher wiedergibt: Kratzig und verzerrt zwar, manche Worte bisweilen unverständlich, aber für markige Mitteilungen wie „Fehler" oder „Finger weg!" reicht's allemal. Die gleiche Firma bietet auch ein Spracherkennungs-Programm an, das der Autor allerdings noch nicht erprobte.

Eine feine Sache auch die Analog-Digital-Wandler im Euro-Apple, mit denen sich – über ein paar Bauteile wie Opto-Koppler oder FETs zum Beispiel – der 2020 bequem Informationen aus elektrischen Systemen beliebiger Art beschaffen kann. Faszinierend schließlich noch, wie der Euro-Apple zum „Software-Basteln" herausfordert: Die vorzügliche Dokumentation, die komfortable Unterstützung durch Monitor und Mini-Assembler verleiten förmlich dazu, sich eigene Mini-Spezial-Hochsprachen zu basteln oder Sonderwünsche, ins Betriebs-System gepflanzt, zu verwirklichen. Um's kurz zu machen: Nach zehn Wochen täglicher (Feierabend- und Wochenend-)Arbeit ist der Autor des Beitrages begeistert von dieser Maschine, die der Programmierer-Fantasie erst sehr viel später Grenzen setzt, als das die billigeren Mitbewerber des 2020 tun (*Bild 3*).

Empfehlenswert – aber für wen?

Nun, wer das Programmieren erst zum Steckenpferd machen will, noch nicht über nennenswerte Erfahrungen verfügt und in den kommenden Jahren nicht sehr viel mehr an Freizeit ins Hobby investieren kann, als einem Vierzig-Wochenstunden-Arbeitnehmer gemeinhin übrigbleibt, der wird – wenn überhaupt – erst sehr spät in die Lage kommen, die spezifischen 2020-Vorteile voll nutzen zu können. Der ist mit Maschinen wie dem PET oder TRS-80 besser bedient als mit dem rund zweimal teureren ITT-Rechner. Das gleiche gilt auch für professionelle Anwender, die mit dem Rechner busorientierte Systeme kontrollieren wollen – da ist der PET mit seinem IEC-Bus bisher unerreicht.

Wo jedoch fortgeschrittenere Stekkenpferd-Reiter am Werke sind oder wo man sich bei professionellen Anwendungen noch nicht auf den IEC-Bus festgelegt hat und dabei neben Basic weitere Hochsprachen zu benutzen wünscht, dort wird der Euro-Apple sicherlich das Rennen machen, wenn ihm nicht die Konkurrenz in die Quere kommt; in den USA bietet Microsoft nämlich für den TRS-80 bereits FORTRAN an.

Eine Morseschreibmaschine

Über den Sinn von Morsedecodern wollen wir uns hier nicht den Kopf zerbrechen; schließlich hat Telegrafie heute nur noch dort Sinn, wo es darum geht, aus einem hohen Störpegel noch ein schwaches Signal herauszuhören, und da kann der beste Morsedecoder nicht mehr mit. Sinnvoll ist es dagegen, Morsezeichen vom Terminal aus zu schreiben. Dazu dient hier der Mikrocomputer KIM-1 zusammen mit einem ASCII-Terminal.

Das hier beschriebene 6502-Maschinencode-Programm setzt jeden von einem ASCII-Terminal aus getippten Buchstaben in das entsprechende Morsezeichen um. Über einen I/O-Port des Mikrocomputers KIM-1 kann direkt z. B. ein Kurzwellensender angesteuert werden; wahlweise ist auch eine Ausgabe als Nf-Ton möglich. Die Eingabe kann sehr schnell erfolgen, da sich der Mikrocomputer einen 255 Zeichen umfassenden Zwischenspeicher aufbaut und diesen mit einer beliebig programmierbaren Geschwindigkeit als Telegrafie (CW) ausgibt.

Bild 1 zeigt das komplette Programm. Es verwendet die in Heft 7/1979, Seite 396, bereits näher besprochene Interrupt-Routine zum Empfang der ASCII-Zeichen, die hier nur innerhalb des „Hex-Dump" wiedergegeben ist. Bild 1 enthält auch die Tabelle zur Umwandlung der ASCII- in Morsezeichen (Adressen 006B...00AA). Es ist zu beachten, daß unbedingt die NMI-Leitung des KIM-1 mit dem Port PB 7 zu verbinden ist, da sonst kein Zeichen vom Terminal empfangen werden kann.

Das Kernstück der Morseschreibmaschine ist der in *Bild 2* gezeigte Programmteil. Er gibt ein im Akkumulator des Mikroprozessors 6502 stehendes

```
0000 A9 02 85 E6 85 E7 85 EB 85 EC 8D 0F 17 A9 00 8D
0010 FB 17 A9 AB 8D FA 17 A5 E6 38 E5 EB F0 F9 C6 E6
0020 A0 00 B1 E6 C9 20 30 EF 29 3F AA BD 6B 00 0A D0
0030 09 A5 E9 85 F4 20 54 00 F0 DD A2 01 90 02 A2 03
0040 86 F4 20 4F 00 A2 01 86 F4 20 54 00 18 90 DF A2
0050 01 8E 01 17 A6 EE A4 ED 88 D0 FD EE 00 17 CA D0
0060 F5 C6 F4 D0 EF A2 00 8E 01 17 60 4C 60 88 A8 90
0070 40 28 D0 08 20 78 B0 48 E0 A0 F0 68 D8 50 10 C0
0080 30 18 70 98 B8 C8 AC 80 54 80 80 80 56 80 80 80
0090 80 80 80 80 80 80 80 CE 8C 56 94 FC 7C 3C 1C 0C
00A0 04 84 C4 E4 F4 56 56 80 80 80 32 48 98 48 A5 E8
00B0 D0 16 2C 40 17 10 09 A9 40 8D 0E 17 68 A8 68 40
00C0 A9 09 85 E8 A9 A0 D0 F1 C6 E8 D0 0E A5 EA F0 06
00D0 C6 EB A0 00 91 EB A9 8E D0 DF 18 2C 40 17 10 01
00E0 38 66 EA 18 90 F0
```

Bild 1. Mit diesem kleinen Programm wird aus dem Mikrocomputer KIM-1 eine „Morseschreibmaschine"

```
0028 29 3F     AND #3F
002A AA        TAX
002B BD 6B 00  LDA 006B,X
002E 0A        ASL A
002F D0 09     BNE 003A
0031 A5 E9     LDA E9
0033 85 F4     STA F4
0035 20 54 00  JSR 0054
0038 F0 DD     BEQ 0017
003A A2 01     LDX #01
003C 90 02     BCC 0040
003E A2 03     LDX #03
0040 86 F4     STX F4
0042 20 4F 00  JSR 004F
0045 A2 01     LDX #01
0047 86 F4     STX F4
0049 20 54 00  JSR 0054
004C 18        CLC
004D 90 DF     BCC 002E
004F A2 01     LDX #01
0051 8E 01 17  STX 1701
0054 A6 EE     LDX EE
0056 A0 80     LDY #80
0058 8C 00 17  STY 1700
005B 88        DEY
005C D0 FD     BNE 005B
005E CA        DEX
005F D0 F5     BNE 0056
0061 C6 F4     DEC F4
0063 D0 EF     BNE 0054
0065 A2 00     LDX #00
0067 8E 01 17  STX 1701
006A 60        RTS

00E9 03 PAUSE
00ED 80 TONFREQUENZ
00EE 60 GESCHWINDIGKEIT
```

Bild 2. Disassemblierte Auflistung des Programmteils für die Ausgabe eines CW-Zeichens zum Tasten eines Senders. Das im Akku stehende ASCII-Zeichen wird in den Morsecode umgesetzt

ASCII-Zeichen in Telegrafie am Port PA 0 aus; dies geschieht als digitales Signal, wobei H-Pegel an PA 0 dem Zustand „Sender aus" und L-Pegel dem Zustand „Sender ein" entspricht.

Für Test- und Übungszwecke kann es sinnvoll sein, nicht ein digitales Signal zum Tasten des Senders zu erzeugen, sondern eine Tonfrequenz, die dann z. B. mit Hilfe der im KIM-1-Handbuch beschriebenen Schaltung (PNP-Transistor als Lautsprecher-Treiber) hörbar gemacht werden kann. Dazu sind die in *Bild 3* angegebenen Adressen zu ändern.

```
0056 A4 ED     LDY ED
0058 88        DEY
0059 D0 FD     BNE 0058
005B EE 00 17  INC 1700
```

Bild 3. Programmänderung zum Erzeugen eines Nf-Tones am Port PA 0

Die Eigenschaften der abgegebenen CW-Zeichen können durch Ändern bestimmter Zero-Page-Adressen den individuellen Erfordernissen angepaßt werden. Die Standard-Inhalte dieser Zellen sind ebenfalls in Bild 2 angegeben. Zu bedenken ist, daß eine Änderung der Tonfrequenz auch die CW-Geschwindigkeit beeinflußt: Höhere Zahlenwerte ergeben niedrigere Frequenzen und Geschwindigkeiten.

Für Übungszwecke wird manchmal der Zeichenabstand über die Norm von drei Punktlängen hinaus verlängert. Dies kann hier durch Einschreiben der gewünschten Punktlängen in die Zelle 00E9 geschehen.

Der von dem Programmteil in Bild 3 erzeugte Nf-Ton hört sich leider ein wenig rauh an. Dies hängt damit zusammen, daß er im Abstand von einigen Millisekunden jeweils für die Dauer einiger Mikrosekunden von der ASCII-Eingabe-Interruptroutine unterbrochen wird. Ein absolut sauberer Ton ist daher nur zu erzeugen, indem man wie in Bild 2 ein rein digitales Signal erzeugt und damit einen Tongenerator steuert, wenn auch die „Unsauberkeit" des Software-Tones nur gering ist.

Der Vorteil einer Morseschreibmaschine wird beim praktischen Betrieb schnell deutlich: Man kann so schnell geben wie bei RTTY, und die Zeichen werden mit konstant exaktem Punkt-Strich- und Punkt-Pausen-Verhältnis gesendet; und das ist wiederum Voraussetzung für das Funktionieren von Morsedecodern – siehe oben.

Herwig Feichtinger

Ist Maschinensprache überholt?

Ganz grob lassen sich die heutigen Mikrocomputer in zwei Klassen einteilen: Auf der einen Seite die BASIC-Rechner, die sich für mathematische Aufgaben und auch ein wenig für kleine Textverarbeitungsprobleme einsetzen lassen, und auf der anderen Seite die „maschinenorientierten" Mikrocomputer, die in erster Linie als Hardware-Ersatz gedacht sind.

In letzter Zeit hörte man häufig die Ansicht, daß letztere Gruppe, die sich „nur" in Maschinensprache bzw. in der Assembler-Sprache programmieren läßt, überholt sei. Ist das wirklich so?

Tatsächlich ist die Programmierung in Maschinensprache mit einigen Nachteilen verbunden. Erstens ist sie systembezogen, d. h. ein Programm, das auf einem bestimmten Mikrocomputer-System läuft, muß für ein anderes erst umgeschrieben werden; wenn dabei unterschiedliche Mikroprozessortypen verwendet werden, ist dies oft sogar unmöglich. Zweitens enthält die Maschinensprache, die der Mikroprozessor ja direkt verarbeitet, nur sehr primitive arithmetische Befehle, z. B. für das Addieren und Subtrahieren zweier Zahlen zwischen 0 und 255. Drittens ist das Erlernen der Maschinensprache ebenso wie die Programmierung zeitraubender als in einer höheren Programmiersprache wie BASIC.

Also wäre es doch besser, wenn wir uns nur noch der höheren Programmiersprachen bedienten?! Nein, ganz so einfach ist das auch wieder nicht. Hier ein Beispiel, bei dem kein vernünftiger Mensch auf die Idee kommen wird, die Maschinensprache durch eine höhere Sprache zu ersetzen.

Die in der FUNKSCHAU beschriebene Mikrocomputer-Uhr zum Decodieren der vom Sender DCF 77 ausgestrahlten Zeitinformation muß natürlich in Echtzeitbetrieb arbeiten; d. h. das Programm muß die empfangene Information sofort in den zur Anzeige notwendigen Code umformen. Dies ist eine ausgesprochen zeitkritische Aufgabe. Ein BASIC-Interpreter, der allein für einen Programmbefehl mehr als tausendmal länger braucht als ein Maschinenbefehl, wäre hier fehl am Platze – ganz abgesehen davon, daß der Hardware-Aufwand wegen der dann notwendigen PROMs mit dem BASIC-Interpreter um mindestens 100 DM steigen würde. Ähnlich ist es bei den jetzt recht beliebten Schachcomputern.

Viele andere auch in der FUNKSCHAU veröffentlichte Programme sind Beispiele dafür, daß man mit BASIC zwar besser rechnen kann, aber daß die Maschinensprache für bestimmte Zwecke unumgänglich ist.

Der oft heraufbeschworene Gegensatz zwischen BASIC- und Assembler-Computern existiert meist nicht; denn Computer wie der PET 2001 oder der TRS-80 lassen sich – wenn auch im Urzustand extrem umständlich – in Maschinensprache programmieren, und Mikrocomputer wie der AIM-65, der SYM-1 oder der KIM-1 lassen sich problemlos mit einem BASIC-Interpreter erweitern, wodurch allerdings ihr Preisvorteil gegenüber BASIC-Computern verlorengeht.

KIM als Siebensegment-Fernschreiber

Fernschreiber sind meist ebenso laute wie unförmige papierverschwendende Geräte. Wenn man auf eine Verewigung empfangener Fernschreibsendungen verzichtet – was besonders im Amateurfunk oft vertretbar ist – so läßt sich das Siebensegment-Display des Mikrocomputers KIM-1 dazu mißbrauchen, RTTY mitzulesen.

Ausgangspunkt des hier vorgestellten Programms ist das im HOBBY-COMPUTER-Sonderheft des Franzis-Verlages abgedruckte Siebensegment-Pseudoalphabet für den KIM-1. Es stellt den Inhalt von sechs aufeinanderfolgenden Speicherzellen als alphanumerischen Siebensegment-Code auf dem sechsstelligen KIM-Display dar.

```
0000 A9 00     LDA #00
0002 85 F3     STA F3
0004 A9 08     LDA #08
0006 85 F4     STA F4
0008 20 60 00  JSR 0060
000B A9 08     LDA #08
000D 2C 02 17  BIT 1702
0010 D0 F2     BNE 0004
0012 C6 F4     DEC F4
0014 10 F2     BPL 0008
0016 A9 05     LDA #05
0018 85 F5     STA F5
001A A9 13     LDA #13
001C 85 F4     STA F4
001E 20 60 00  JSR 0060
0021 C6 F4     DEC F4
0023 10 F9     BPL 001E
0025 18        CLC
0026 A9 08     LDA #08
0028 2C 02 17  BIT 1702
002B F0 01     BEQ 002E
002D 38        SEC
002E 66 F0     ROR F0
0030 C6 F5     DEC F5
0032 10 E6     BPL 001A
0034 A5 F0     LDA F0
0036 6A        ROR A
0037 6A        ROR A
0038 29 1F     AND #1F
003A C9 1F     CMP #1F
003C F0 C2     BEQ 0000
003E C9 1B     CMP #1B
0040 D0 04     BNE 0046
0042 A9 20     LDA #20
0044 D0 BC     BNE 0002
0046 05 F3     ORA F3
0048 AA        TAX
0049 BD A0 00  LDA 00A0,X
004C 48        PHA
004D A2 04     LDX #04
004F B5 E0     LDA E0,X
0051 95 E1     STA E1,X
0053 CA        DEX
0054 10 F9     BPL 004F
0056 68        PLA
0057 85 E0     STA E0
0059 4C 04 00  JMP 0004
```

Bild 1. Programmteil zum Empfang von Baudot-Fernschreibzeichen. Die Werte an den Adressen 0005 (halbe Startbitlänge) und 001B (Bitlänge) sind für die Baudrate maßgebend (hier 50 Bd)

Das Programm (*Bild 1*) zum Empfang der Fernschreibzeichen geht davon aus, daß an Port PB 3 ein Empfangskonverter angeschlossen ist, der dem Mark-Signal den H-Pegel (TTL) und dem Space-Signal den Low-Pegel zuordnet. Geeignete Schaltungen hierfür finden sich in FUNKSCHAU 1978, Heft 14, und in FUNKSCHAU 1979, Heft 9, wobei lediglich u. U. durch Zwischenschalten eines Inverters die richtige Polarität herzustellen ist.

Eine Besonderheit des Programms ist, daß die Zeichen nur dann ausgewertet werden, wenn das Startbit mindestens die halbe Datenbitlänge aufweist. Dadurch werden kurzzeitige Störimpulse wirkungsvoll unterdrückt.

Die schon erwähnte Display-Routine wurde hier lediglich als Hex-Liste abgedruckt (*Bild 2*). Sie wurde nur geringfügig geändert, so daß sie rund 1 ms dauert und dadurch auch als „Timer" zur Verzögerung definierter Zeiten verwendet werden kann. Der Adressenbereich 00A0...00DF enthält eine Umsetztabelle vom Baudot-Code in den Siebensegment-Code; die Reihenfolge der Zeichen entspricht darin der in ELEKTRONIK 1978, Heft 14, Seite 62 abgedruckten Tabelle, so daß Änderungen leicht möglich sind.

Sicher ist das Pseudoalphabet bei der ersten Inbetriebnahme des Fernschreibprogramms etwas ungewohnt. Der Text läuft mit der vollen Geschwindigkeit der empfangenen Zeichen von rechts nach links durch. Es besteht aber kein Grund, sich anfangs von dem „lustigen Flackern" einschüchtern zu lassen. Bei einiger Übung im Lesen des Pseudoalphabets – Cyrillisch kann man ja schließlich auch erlernen – ist das Mitlesen auch bei vollen 45 oder 50 Baud noch kein Problem. Bei Amateur-Funkfernschreiben (z. B. im KW-Bereich oder 2-m-Band) liegt die durchschnittliche Geschwindigkeit ohnehin meist niedriger.

Es sei noch erwähnt, daß auch hier im Prinzip eine Geschwindigkeitsumsetzung mittels eines Bufferspeichers möglich wäre, wie dies in FUNKSCHAU 1979, Heft 7, unter dem Titel „ASCII-Eingabe – ganz nebenbei" beschrieben wurde. In der Praxis hat sich dies beim Empfang von u. U. gestörten Fernschreibsendungen jedoch nicht bewährt, da man die Auswirkungen einer Falschabstimmung des Empfängers erst nach einigen Sekunden auf dem Display zu Gesicht bekommt, was die Abstimmung erheblich erschwert.

Gerhard Kirchberger
Herwig Feichtinger

```
0060 A9 7F 8D 41 17 A2 09 A0 06 B9 DF 00 84 FC A0 00
0070 8C 40 17 8E 42 17 8D 40 17 A0 18 88 D0 FD E8 E8
0080 A4 FC 88 D0 E4 8E 42 17 A9 00 8D 41 17 60 00 00

00A0 80 F9 80 F7 80 ED 85 BE 80 DE D0 9E D4 F1 D8 F8
00B0 B1 DB B8 FE F4 EE F3 E7 DC FC BD 80 B7 F6 9C 80
00C0 80 CF 80 80 80 80 FF 87 80 80 E6 80 80 80 C1 80
00D0 E6 80 80 DB 80 FD BF 86 EF D3 80 80 80 80 C1 80

00F0 ZEICHENBUFFER
00F3 BU/ZI-FLAG
00F4 ZEITGEBER
00F5 BITZAEHLER
00E0...00E5 DISPL.
```

Bild 2. Das Display-Unterprogramm (0060...008D) und die Baudot-Siebensegment-Umsetztabelle (00A0...00DF)

Hans Werner

Mikrocomputer

Eine kleine Einführung

21. Teil

Schieben

Bei allen Mikrocomputern kann der Inhalt des Akkumulators um eine oder mehrere Stellen verschoben werden.

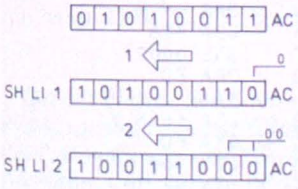
Schiebebefehle können sehr nützlich sein, z. B. um auf einfache Art Multiplikationen mit 2, 4, 8 usw. auszuführen. Beispiel:

```
7·2 = 14        0 0 0 0 0 1 1 1    7
                0 0 0 0 1 1 1 0    14
```

Nach links schieben

SH LI n bedeutet: Schiebe den Inhalt des Akkus um n Stellen nach links.

Die Befehle SH LI 1 und SH LI 2 verdeutlichen den Vorgang. Freigewordene Stellen werden mit Nullen aufgefüllt, die überflüssigen Bits höherer Wertigkeit entfallen.



Das Carry-Flag wird auf 0, das Vorzeichen V und die Nullmeldung Z entsprechend dem Akkuinhalt gesetzt.

```
Vor Befehl SH LI 1:  C = ?  V = 0  Z = 0
Nach       SH LI 1:  C = 0  V = 1  Z = 0
```

Nach rechts schieben

SH RE n bedeutet: Schiebe den Inhalt des Akkus um n Stellen nach rechts. Die höherwertigen Stellen werden mit Nullen aufgefüllt. Das Statusregister verhält sich wie bei dem Befehl SH LI.

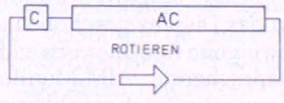
Beispiel:

Inhalt des Akkus: 10111001

nach SH RE 4: 00001011

Abarten:

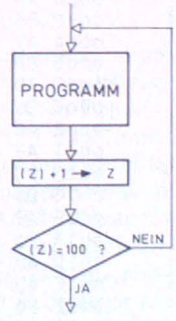
Einige Mikrocomputer beziehen das Carry-Bit mit in den Schiebevorgang



ein. Bei anderen rotiert der Akkuinhalt, d. h. die höherwertigen Stellen erscheinen wieder links im Akku und umgekehrt.

Inkrementieren

Ist z. B. ein Programm 100mal zu durchlaufen, wird normalerweise der Inhalt einer bestimmten Speicherzelle Z in jedem Durchlauf um 1 erhöht und beim Erreichen der Zahl 100 die Programmschleife verlassen. Dazu muß der Inhalt von Z geladen, eine 1 addiert und das Ergebnis wieder in Z gespeichert werden. Einfacher geht es mit dem Befehl:



IN n Inkrementiere den Inhalt der Speicherzelle n; anders ausgedrückt: Erhöhe den Inhalt der Zelle n um 1.

Dekrementieren

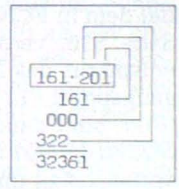
Mit dem Befehl Dekrementieren wird der Inhalt der adressierten Speicherzelle um 1 verringert.

DEKREMENT = Minderung

DE n Dekrementiere den Inhalt der Speicherzelle n

Multiplikation

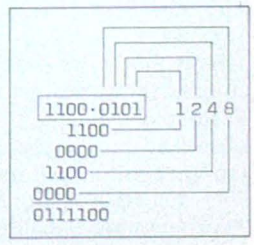
Mikrocomputer können meist nicht multiplizieren. Deshalb gehört die Multiplikation zu den wichtigsten Grundprogrammen. Jede Multiplikation kann durch fortlaufende Addition gelöst werden, doch führt dieser Weg zu Programmen mit langer Laufzeit.



Eine andere Möglichkeit ist die bekannte schriftliche Multiplikation von Dezimalzahlen. Der Multiplikand wird in Einer, Zehner usw. zerlegt und die sich ergebenden Teilprodukte 161·1, 161·0, 161·2 addiert. Die Teilprodukte werden jeweils um eine Stelle nach links eingerückt, damit die richtige Wertigkeit (Einer, Zehner...) erhalten bleibt. Das Verfahren läßt sich ohne Änderung auf das Binärsystem übertragen.

```
Beispiel:  12·5 = 60
1. 1100·1 =    1100
2. 1100·0 =    0000
3. 1100·1 =   1100
4. 1100·0 = 0000
```

Vor der Addition wird wiederum jedes Teilergebnis um eine Stelle nach links eingerückt.



```
      0
+1100 ------ Multiplikanden-Bit = 1     (1)
   1100
+0000 ------                 = 0         (2)
  01100
+1100 ------                 = 1         (4)
 111100
+0000 ------                 = 0         (8)
0111100
```

Im dualen System benötigt man lediglich ein „Mikro-Einmaleins", da der Multiplikator nur mit 0 oder 1 zu multiplizieren ist. Schematisch kann die Aufgabe durch 4 Additionen gelöst werden. Abhängig vom Multiplikanden-Bit 1, 2, 3, 4, 8 wird entweder der Multiplikator oder Null addiert. Im Beispiel ist das 1100 bzw. 0000. Nach jeder Addition muß der Multiplikator um eine Stelle nach links geschoben werden.

Multiplikationen mit 8 bit langen Zahlen benötigen auch entsprechend 8 Additionen.

Multiplikations-Programm

Programmbeispiele müssen nicht auf die ideale Form gebracht werden, sie sollen leicht durchschaubar und verständlich sein. Deshalb sind 2 Einschränkungen notwendig:

1. Jeder Faktor darf nicht länger als 4 bit sein.

2. Vorzeichen werden nicht berücksichtigt.

Das höchste Ergebnis ist 1111: 1111 = 11100001 (225). Ein Überlauf des Akkumulators ist also nicht zu erwarten. Bei größeren Faktoren kann das Ergebnis 1 Byte überschreiten. Das erfordert eine umständliche Speicherplatzreservierung und ist für die prinzipielle Erklärung unwichtig.

Fortsetzung folgt

AIM-65 empfängt RTTY

Das nachfolgende 6502-Programm läuft ohne Änderung sowohl auf dem AIM-65 von Rockwell als auch auf dem PC-100 von Siemens. Es stellt empfangene Fernschreibzeichen als durchlaufende Schrift auf dem 20stelligen alphanumerischen Display dar.

Zu diesem Zweck ist der Ausgang eines RTTY-(Fernschreib-)Konverters mit dem Port PA 0 bzw. dem Pin 14 des Application Connector zu verbinden. Bei dem in FUNKSCHAU 1979, Heft 9, Seite 530, veröffentlichten Konverter entspricht das Mark-Signal dem Ausgangspegel L; im Programmlisting (*Bild*) sind dann die Befehle CLC und SEC an den Adressen 0026 und 002C zu vertauschen, und an die Adresse 0016 ist BEQ (hex F0) zu schreiben.

Je nach Geschwindigkeit der empfangenen Zeichen sind bestimmte Speicherzellen mit passenden Zeitkonstanten zu belegen, nämlich 0019 und 0020. Bei 75 Bd sind es die Werte 1A und 34, bei 50 Bd 28 und 50 und bei 45 Bd 2A und 57. Zusätzlich zum Programm werden Speicherzellen im Zero-Page-Bereich belegt, und zwar 00E8...00FC als Displaybuffer, 00FD als Flag für die Buchstaben-Ziffern-Umschaltung im Baudot-Code, 00FE als Buffer für das gerade empfangene Zeichen und 00FF als X-Zwischenspeicher. Die Startadresse des Programms ist 0000.

Beim Start wird das Flag zunächst auf „Buchstaben" gesetzt und die Escape-Taste des AIM-65 abgefragt. Mit ihr kann man das Programm wieder verlassen, hier mit einem indirekten Sprung über den RST-Vektor zum AIM-Monitorprogramm. Sobald ein Startbit erkannt wird, geschieht die Serien-Parallel-Wandlung per Software. Das empfangene Zeichen wird dann mit der an der Adresse 0068 stehenden Tabelle in den ASCII-Code umgewandelt und von rechts in das Display geschoben. Dafür wird eine eigene Routine verwendet, da das entsprechende AIM-Monitor-Unterprogramm die Zeichenzahl begrenzt.

Die Monitor-Routine an der Adresse EC18 dient zur Verzögerung um (256·A) µs, diejenige bei EF7B gibt ein ASCII-Zeichen an der durch X spezifizierten Display-Stelle aus. Will man einen anderen Port als PA 0 als Eingang verwenden, so braucht man nur die PA-Maske zu ändern; sie steht an den Adressen 0012 und 0025. Fe.

```
0000 A9 00     LDA #00
0002 85 FD     STA FD
0004 A9 FB     LDA #FB
0006 8D 80 A4  STA A480
0009 AD 82 A4  LDA A482
000C 30 03     BMI 0011
000E 6C FC FF  JMP (FFFC)
0011 A9 01     LDA #01
0013 2C 0F A0  BIT A00F
0016 D0 EC     BNE 0004
0018 A9 1A     LDA #1A
001A 20 18 EC  JSR EC18
001D A0 06     LDY #06
001F A9 34     LDA #34
0021 20 18 EC  JSR EC18
0024 A9 01     LDA #01
0026 18        CLC
0027 2C 0F A0  BIT A00F
002A F0 01     BEQ 002D
002C 38        SEC
002D 66 FE     ROR FE
002F 88        DEY
0030 D0 ED     BNE 001F
0032 A5 FE     LDA FE
0034 6A        ROR A
0035 6A        ROR A
0036 29 1F     AND #1F
0038 F0 C6     BEQ 0000
003A C9 1F     CMP #1F
003C F0 C2     BEQ 0000
003E C9 1B     CMP #1B
0040 D0 04     BNE 0046
0042 A9 20     LDA #20
0044 D0 BC     BNE 0002
0046 05 FD     ORA FD
0048 AA        TAX
0049 BD 68 00  LDA 0068,X
004C 85 FC     STA FC
004E A2 00     LDX #00
0050 86 FF     STX FF
0052 B5 E9     LDA E9,X
0054 95 E8     STA E8,X
0056 09 80     ORA #80
0058 20 7B EF  JSR EF7B
005B A6 FF     LDX FF
005D E8        INX
005E E0 14     CPX #14
0060 D0 EE     BNE 0050
0062 F0 A0     BEQ 0004

0068 00 45 0A 41 20 53 49 55
0070 0D 44 52 4A 4E 46 43 4B
0078 54 5A 4C 57 48 59 50 51
0080 4F 42 47 00 4D 58 56 00
0088 44 33 00 2D 20 27 38 37
0090 00 00 34 3B 2C 5B 3A 28
0098 35 2B 29 32 21 36 30 31
00A0 39 3F 5D 00 2E 2F 3D 00
```

Funkfernschreibsendungen im Baudot-Code können mit diesem Programm auf den Displays der Mikrocomputer AIM-65 und PC-100 als durchlaufende Schrift dargestellt werden

Der Mikrocomputer TM 990/189

Vorwiegend für Ausbildungszwecke ist der jüngst von Texas Instruments auf den Markt gebrachte Ein-Platinen-Computer TM 990/189 nach Aussage des Herstellers gedacht. Er besitzt die schon fast minicomputer-ähnliche 16-bit-CPU TMS 9980, die bereits multiplizieren und dividieren kann, und gehört damit zu den heute interessantesten Single-Board-Computern.

Hardware...

Der Computer besteht aus einer 28 cm x 20,5 cm großen Epoxy-Platine mit rund 30 ICs. Eine alphanumerische Eingabetastatur ist zusammen mit einer zehnstelligen Siebensegmentanzeige in einem der Oberseite des Taschenrechners TI-58 gleichenden Gehäuse untergebracht, das fest auf die Platine montiert ist. Die Siebensegment-Anzeige gestattet im Prinzip auch die Anzeige beliebiger Buchstaben und Zeichen, die allerdings oft recht seltsam aussehen, wie etwa M, W, X usw., die sich aber nach einiger Übung sicher beherrschen lassen. Ein kleiner piezoelektrischer Lautsprecher bestätigt jeden Tastendruck mit einem kurzen Pfeifton – beruhigend, wenn auch nicht unbedingt erforderlich, da die Tasten alle einen fühlbaren Druckpunkt besitzen.

Auf der Platine befinden sich 4 KByte ROM mit dem Monitorprogramm, das die Tastenabfrage und die Anzeige von Adressen und Daten bewirkt, und 1 KByte RAM, das sich dank freier Sockel ohne Lötarbeiten auf 2 KByte erweitern läßt. Für den Anschluß eines Kassettenrecorders für die Speicherung von Programmen und Datenblöcken ist die Platine schon bestückt, dagegen muß sie für die Verbindung mit einem externen Terminal (RS-232-Schnittstelle) erst nachgerüstet werden.

Tabelle 1 zeigt die vom Benutzer verwendbaren „Extended Operations", und die *Tabelle 2* die Speicherbelegung auf der Mikrocomputer-Platine. Wie wir noch sehen werden, sind der Erweiterung leider enge Grenzen gesetzt: Der Mikroprozessor kann nämlich nur insgesamt 16 KByte adressieren.

... und Software

Schon in der Grundversion besitzt der Mikrocomputer TM 990/189 etwas, was man bei anderen erst für teures Geld zusätzlich erwerben muß – nämlich einen symbolischen Assembler. Er sorgt dafür, daß die Programmbefehle nicht in hexadezimaler Form eingegeben werden müssen; es genügt die leichter verständliche mnemonische Form. „Symbolisch" bedeutet, daß keine absoluten Adressen im Assembler-Programm verwendet werden brauchen, sondern nur Marken, sprich „Labels", die der Assembler selbst in Adressen umrechnet. Dies ist besonders dann nützlich, wenn man Programmteile verschiebt oder zusätzliche Befehle einfügt, wodurch sich alle absoluten Adressen ändern würden.

Die Software-Eigenschaften eines Mikrocomputers werden selbstverständlich in erster Linie vom Befehlsvorrat der verwendeten CPU bestimmt. Hier handelt es sich um den Typ TMS 9980, der eine Art abgemagerte Version des TMS 9900 darstellt. Die Abmagerung bezieht sich jedoch nur auf die Hardware: Die sechzehn Datenbusleitungen wurden z. B. mit Hilfe einer internen Mulitplex-Schaltung auf acht reduziert, um Gehäusepins und damit Kosten zu sparen. Geblieben sind dagegen solche Dinge wie etwa eine Hardware-Logik zur Seriell/Parallel-Umwandlung für den externen Datenverkehr (CRU) und viele andere Besonderheiten, die man nur bei der 9900-Familie findet. Sehen wir uns also die CPU einmal genauer an.

Die CPU 9980

Die CPU TMS 9980 ist ein 16-bit-Mikroprozessor in NMOS-Technik mit 14 Adressenleitungen (deswegen ist der ansprechbare Adressenbereich nur 16 KByte) und sechs Interrupts. Die 16-bit-Datenworte werden von einem Multiplexer in zwei 8-bit-Teile gespalten und nacheinander auf den 8-bit-Datenbus gegeben. Natürlich vermindert das Geschwindigkeit und Datendurchsatz der CPU gegenüber „echten" 16-bit-Prozessoren, vermindert aber die Anzahl der Gehäusepins auf 40 und sorgt damit für geringere Hardware-Kosten als bei der „Mutter-CPU" TMS 9900.

Wesentlichstes Merkmal der CPU ist, daß sie keinen Akku im herkömmlichen Sinne enthält. Arithmetische Operationen können vielmehr an frei programmierbaren Adressen des System-RAM ausgeführt werden, ohne den Umweg über eine solche „Krücke" zu machen. Diese RAM-Adressen dienen praktisch als CPU-externe Register. Ein Datentransfer ist also direkt von Speicherplatz zu Speicherplatz möglich.

Tabelle 1: Monitor-Unterprogramme

XOP	Funktion
8	Hex-Zeichen auf dem Terminal drucken
9	Hex-Zeichen vom Terminal holen
10	Vier Hex-Zeichen auf dem Terminal drucken
11	Kombination aus XOP 13 und XOP 12
12	ASCII-Zeichen ausgeben
13	ASCII-Zeichen vom Terminal holen
14	Text auf dem Terminal ausgeben

Tabelle 2: Speicherbelegung

0000...007F Interrupt- und XOP-Vektoren
0080...0145 Monitor-System-RAM
0146...01FF Assembler-Symboltabelle
0200...03FF Anwender-RAM
0400...0FFF Erweiterungen
3000...3FFF Betriebssystem-ROM

```
      0
  +1100 ------ Multiplikanden-Bit = 1    (1)
   1100
  +0000 ------                 = 0       (2)
  01100
  +1100 ------                 = 1       (4)
  111100
  +0000 ------                 = 0       (8)
 0111100
```

Im dualen System benötigt man lediglich ein „Mikro-Einmaleins", da der Multiplikator nur mit 0 oder 1 zu multiplizieren ist. Schematisch kann die Aufgabe durch 4 Additionen gelöst werden. Abhängig vom Multiplikanden-Bit 1, 2, 3, 4, 8 wird entweder der Multiplikator oder Null addiert. Im Beispiel ist das 1100 bzw. 0000. Nach jeder Addition muß der Multiplikator um eine Stelle nach links geschoben werden.
Multiplikationen mit 8 bit langen Zahlen benötigen auch entsprechend 8 Additionen.

Multiplikations-Programm

Programmbeispiele müssen nicht auf die ideale Form gebracht werden, sie sollen leicht durchschaubar und verständlich sein. Deshalb sind 2 Einschränkungen notwendig:

1. Jeder Faktor darf nicht länger als 4 bit sein.

2. Vorzeichen werden nicht berücksichtigt.

Das höchste Ergebnis ist 1111: 1111 = 11100001 (225). Ein Überlauf des Akkumulators ist also nicht zu erwarten. Bei größeren Faktoren kann das Ergebnis 1 Byte überschreiten. Das erfordert eine umständliche Speicherplatzreservierung und ist für die prinzipielle Erklärung unwichtig.

Fortsetzung folgt

AIM-65 empfängt RTTY

Das nachfolgende 6502-Programm läuft ohne Änderung sowohl auf dem AIM-65 von Rockwell als auch auf dem PC-100 von Siemens. Es stellt empfangene Fernschreibzeichen als durchlaufende Schrift auf dem 20stelligen alphanumerischen Display dar.

Zu diesem Zweck ist der Ausgang eines RTTY-(Fernschreib-)Konverters mit dem Port PA 0 bzw. dem Pin 14 des Application Connector zu verbinden. Bei dem in FUNKSCHAU 1979, Heft 9, Seite 530, veröffentlichten Konverter entspricht das Mark-Signal dem Ausgangspegel L; im Programmlisting (*Bild*) sind dann die Befehle CLC und SEC an den Adressen 0026 und 002C zu vertauschen, und an die Adresse 0016 ist BEQ (hex F0) zu schreiben.

Je nach Geschwindigkeit der empfangenen Zeichen sind bestimmte Speicherzellen mit passenden Zeitkonstanten zu belegen, nämlich 0019 und 0020. Bei 75 Bd sind es die Werte 1A und 34, bei 50 Bd 28 und 50 und bei 45 Bd 2A und 57. Zusätzlich zum Programm werden Speicherzellen im Zero-Page-Bereich belegt, und zwar 00E8...00FC als Displaybuffer, 00FD als Flag für die Buchstaben-Ziffern-Umschaltung im Baudot-Code, 00FE als Buffer für das gerade empfangene Zeichen und 00FF als X-Zwischenspeicher. Die Startadresse des Programms ist 0000.

Beim Start wird das Flag zunächst auf „Buchstaben" gesetzt und die Escape-Taste des AIM-65 abgefragt. Mit ihr kann man das Programm wieder verlassen, hier mit einem indirekten Sprung über den RST-Vektor zum AIM-Monitorprogramm. Sobald ein Startbit erkannt wird, geschieht die Serien-Parallel-Wandlung per Software. Das empfangene Zeichen wird dann mit der an der Adresse 0068 stehenden Tabelle in den ASCII-Code umgewandelt und von rechts in das Display geschoben. Dafür wird eine eigene Routine verwendet, da das entsprechende AIM-Monitor-Unterprogramm die Zeichenzahl begrenzt.

Die Monitor-Routine an der Adresse EC18 dient zur Verzögerung um (256·A) µs, diejenige bei EF7B gibt ein ASCII-Zeichen an der durch X spezifizierten Display-Stelle aus. Will man einen anderen Port als PA 0 als Eingang verwenden, so braucht man nur die PA-Maske zu ändern; sie steht an den Adressen 0012 und 0025. Fe.

```
0000 A9 00     LDA #00
0002 85 FD     STA FD
0004 A9 FB     LDA #FB
0006 8D 80 A4  STA A480
0009 AD 82 A4  LDA A482
000C 30 03     BMI 0011
000E 6C FC FF  JMP (FFFC)
0011 A9 01     LDA #01
0013 2C 0F A0  BIT A00F
0016 D0 EC     BNE 0004
0018 A9 1A     LDA #1A
001A 20 18 EC  JSR EC18
001D A0 06     LDY #06
001F A9 34     LDA #34
0021 20 18 EC  JSR EC18
0024 A9 01     LDA #01
0026 18        CLC
0027 2C 0F A0  BIT A00F
002A F0 01     BEQ 002D
002C 38        SEC
002D 66 FE     ROR FE
002F 88        DEY
0030 D0 ED     BNE 001F
0032 A5 FE     LDA FE
0034 6A        ROR A
0035 6A        ROR A
0036 29 1F     AND #1F
0038 F0 C6     BEQ 0000
003A C9 1F     CMP #1F
003C F0 C2     BEQ 0000
003E C9 1B     CMP #1B
0040 D0 04     BNE 0046
0042 A9 20     LDA #20
0044 D0 BC     BNE 0002
0046 05 FD     ORA FD
0048 AA        TAX
0049 BD 68 00  LDA 0068,X
004C 85 FC     STA FC
004E A2 00     LDX #00
0050 86 FF     STX FF
0052 B5 E9     LDA E9,X
0054 95 E8     STA E8,X
0056 09 80     ORA #80
0058 20 7B EF  JSR EF7B
005B A6 FF     LDX FF
005D E8        INX
005E E0 14     CPX #14
0060 D0 EE     BNE 0050
0062 F0 A0     BEQ 0004

0068 00 45 0A 41 20 53 49 55
0070 0D 44 52 4A 4E 46 43 4B
0078 54 5A 4C 57 48 59 50 51
0080 4F 42 47 00 4D 58 56 00
0088 44 33 00 2D 20 27 38 37
0090 00 00 34 3B 2C 5B 3A 28
0098 35 2B 29 32 21 36 30 31
00A0 39 3F 5D 00 2E 2F 3D 00
```

Funkfernschreibsendungen im Baudot-Code können mit diesem Programm auf den Displays der Mikrocomputer AIM-65 und PC-100 als durchlaufende Schrift dargestellt werden

Der Mikrocomputer TM 990/189

Vorwiegend für Ausbildungszwecke ist der jüngst von Texas Instruments auf den Markt gebrachte Ein-Platinen-Computer TM 990/189 nach Aussage des Herstellers gedacht. Er besitzt die schon fast minicomputer-ähnliche 16-bit-CPU TMS 9980, die bereits multiplizieren und dividieren kann, und gehört damit zu den heute interessantesten Single-Board-Computern.

Hardware...

Der Computer besteht aus einer 28 cm x 20,5 cm großen Epoxy-Platine mit rund 30 ICs. Eine alphanumerische Eingabetastatur ist zusammen mit einer zehnstelligen Siebensegmentanzeige in einem der Oberseite des Taschenrechners TI-58 gleichenden Gehäuse untergebracht, das fest auf die Platine montiert ist. Die Siebensegment-Anzeige gestattet im Prinzip auch die Anzeige beliebiger Buchstaben und Zeichen, die allerdings oft recht seltsam aussehen, wie etwa M, W, X usw., die sich aber nach einiger Übung sicher beherrschen lassen. Ein kleiner piezoelektrischer Lautsprecher bestätigt jeden Tastendruck mit einem kurzen Pfeifton – beruhigend, wenn auch nicht unbedingt erforderlich, da die Tasten alle einen fühlbaren Druckpunkt besitzen.

Auf der Platine befinden sich 4 KByte ROM mit dem Monitorprogramm, das die Tastenabfrage und die Anzeige von Adressen und Daten bewirkt, und 1 KByte RAM, das sich dank freier Sockel ohne Lötarbeiten auf 2 KByte erweitern läßt. Für den Anschluß eines Kassettenrecorders für die Speicherung von Programmen und Datenblöcken ist die Platine schon bestückt, dagegen muß sie für die Verbindung mit einem externen Terminal (RS-232-Schnittstelle) erst nachgerüstet werden.

Tabelle 1 zeigt die vom Benutzer verwendbaren „Extended Operations", und die *Tabelle 2* die Speicherbelegung auf der Mikrocomputer-Platine. Wie wir noch sehen werden, sind der Erweiterung leider enge Grenzen gesetzt: Der Mikroprozessor kann nämlich nur insgesamt 16 KByte adressieren.

... und Software

Schon in der Grundversion besitzt der Mikrocomputer TM 990/189 etwas, was man bei anderen erst für teures Geld zusätzlich erwerben muß – nämlich einen symbolischen Assembler. Er sorgt dafür, daß die Programmbefehle nicht in hexadezimaler Form eingegeben werden müssen; es genügt die leichter verständliche mnemonische Form. „Symbolisch" bedeutet, daß keine absoluten Adressen im Assembler-Programm verwendet werden brauchen, sondern nur Marken, sprich „Labels", die der Assembler selbst in Adressen umrechnet. Dies ist besonders dann nützlich, wenn man Programmteile verschiebt oder zusätzliche Befehle einfügt, wodurch sich alle absoluten Adressen ändern würden.

Die Software-Eigenschaften eines Mikrocomputers werden selbstverständlich in erster Linie vom Befehlsvorrat der verwendeten CPU bestimmt. Hier handelt es sich um den Typ TMS 9980, der eine Art abgemagerte Version des TMS 9900 darstellt. Die Abmagerung bezieht sich jedoch nur auf die Hardware: Die sechzehn Datenbusleitungen wurden z. B. mit Hilfe einer internen Mulitplex-Schaltung auf acht reduziert, um Gehäusepins und damit Kosten zu sparen. Geblieben sind dagegen solche Dinge wie etwa eine Hardware-Logik zur Seriell/Parallel-Umwandlung für den externen Datenverkehr (CRU) und viele andere Besonderheiten, die man nur bei der 9900-Familie findet. Sehen wir uns also die CPU einmal genauer an.

Die CPU 9980

Die CPU TMS 9980 ist ein 16-bit-Mikroprozessor in NMOS-Technik mit 14 Adressenleitungen (deswegen ist der ansprechbare Adressenbereich nur 16 KByte) und sechs Interrupts. Die 16-bit-Datenworte werden von einem Multiplexer in zwei 8-bit-Teile gespalten und nacheinander auf den 8-bit-Datenbus gegeben. Natürlich vermindert das Geschwindigkeit und Datendurchsatz der CPU gegenüber „echten" 16-bit-Prozessoren, vermindert aber die Anzahl der Gehäusepins auf 40 und sorgt damit für geringere Hardware-Kosten als bei der „Mutter-CPU" TMS 9900.

Wesentlichstes Merkmal der CPU ist, daß sie keinen Akku im herkömmlichen Sinne enthält. Arithmetische Operationen können vielmehr an frei programmierbaren Adressen des System-RAM ausgeführt werden, ohne den Umweg über eine solche „Krücke" zu machen. Diese RAM-Adressen dienen praktisch als CPU-externe Register. Ein Datentransfer ist also direkt von Speicherplatz zu Speicherplatz möglich.

Tabelle 1: Monitor-Unterprogramme

XOP	Funktion
8	Hex-Zeichen auf dem Terminal drucken
9	Hex-Zeichen vom Terminal holen
10	Vier Hex-Zeichen auf dem Terminal drucken
11	Kombination aus XOP 13 und XOP 12
12	ASCII-Zeichen ausgeben
13	ASCII-Zeichen vom Terminal holen
14	Text auf dem Terminal ausgeben

Tabelle 2: Speicherbelegung

0000...007F Interrupt- und XOP-Vektoren
0080...0145 Monitor-System-RAM
0146...01FF Assembler-Symboltabelle
0200...03FF Anwender-RAM
0400...0FFF Erweiterungen
3000...3FFF Betriebssystem-ROM

Dazu gehört auch der 2-Adreß-Befehlssatz. (Das stimmt natürlich nur scheinbar – der Transfer erfolgt in Wirklichkeit nach wie vor über ein CPU-internes Datenregister.)

Das „Speicher-zu-Speicher"-Konzept hört sich zwar gut an, hat aber auch Nachteile. Erstens wird ein Mikrocomputer dadurch RAM-intensiv und man muß sich stets genau überlegen, wo man den „Workspace" ohne Behinderung von Programm und Daten definiert. Zweitens geht der Geschwindigkeitsvorteil von Operationen verloren, die innerhalb der Register ausgeführt werden, z. B. Zähl- oder Schiebebefehle, die beim Workspace-Konzept langsamer arbeiten als bei Mikrocomputern, bei denen sich die Register in der CPU selbst befinden.

Die TI-Literatur erwähnt zuweilen den Begriff „Context switch". Darunter versteht man das Umdefinieren dieser Registeradressen auf einen anderen Speicherbereich. Doch sehen wir uns erst einmal an, was das für Register sind.

Hardware- und Software-Register

Die CPU 9980 enthält auf dem Chip nur drei Hardware-Register, nämlich den Program Counter, das Status-Register und – das unterscheidet sie völlig von anderen Prozessoren – einen sog. Workspace Pointer. Er enthält eine bestimmte Adresse, die als Anfangsadresse des „Workspace" dient. Der Workspace ist ein Satz von 16 Registern (R 0...15), die hier eben nicht innerhalb der CPU, sondern im RAM liegen. (Dieses Konzept ähnelt ein wenig der CPU 6502, bei der die „Zero Page" praktisch 256 8-bit- oder 128 16-bit-Register enthält, läßt sich aber doch schlecht vergleichen.)

Der Workspace Pointer macht die Interrupt-Verarbeitung besonders einfach; während der Interrupt-Routine braucht man ihn nur umzudefinieren, und schon werden alle Registerinhalte des Hauptprogramms gerettet.

Was die CPU 9980 noch von anderen Mikroprozessoren (außer vom SCMP) unterscheidet, ist das Fehlen eines Kellerspeichers (Stack) zum Abspeichern der Rücksprungadressen bei Unterprogrammen und Interrupt-Routinen. Vielmehr dienen dazu bestimmte Register im Workspace, z. B. R 11. Selbstverständlich lassen sich hier trotzdem verschachtelte Unterprogrammsprünge durchführen.

Befehlsvorrat

Die CPU 9980 besitzt außer den „üblichen" Lade- und Verschiebebefehlen eine Reihe interessanter Instruktionen, die man sonst kaum findet. So ist etwa JOP (Jump on odd Parity) ebenso unüblich wie SWPB (Swap Bytes), MPY (Multiply) und DIV (Divide). Die Operationscodes sind jeweils 16-bit-Worte. Für die wichtigsten Befehle stehen sieben Adressierungsarten zur Verfügung: Direkte Register-Adressierung, Indirekte Register-Adressierung mit oder ohne automatischer Inkrementierung, indizierte oder nicht indizierte symbolische Speicheradressierung, unmittelbare (immediate) Adressierung und schließlich relative Adressierung mit dem Programmzähler.

Grundsätzlich gibt es 2-Byte- und 4-Byte-Befehle (1 Byte = 8 bit). Es sei noch erwähnt, daß Texas Instruments aus unerfindlichen Gründen die Bezeichnungen für Adressen- und Datenbus „hintenherum" verwendet, d. h. A 0 (Null) ist die höchstwertige Leitung, wovon man sich leicht verwirren läßt.

770 Seiten Papier

Zu Mikrocomputern gehört normalerweise nicht nur die Rechner-Platine selbst, sondern auch eine ganze Menge Papier, die CPU- und Systemeigenschaften erläutert. Leider existiert bisher kein einziger Hersteller, der sich bei dieser Literatur Gedanken über eine didaktische Verwertbarkeit dieses Papierberges gemacht hat – offenbar auch Texas Instruments nur in begrenztem Umfang. Schlägt man den mitgelieferten, rund 195 Seiten starken „User's Guide" auf, so findet man zwar nach einigem Suchen alle Begriffe erklärt, über die irgendwelche Unklarheit bestehen kann, als Lehrbuch ist der User's Guide allerdings ebenso wie die 575 Seiten starke „Introduction to Microcomputers" wohl kaum geeignet. Wie Texas Instruments versprach, wird allerdings bald eine Version vorliegen, die im Gegensatz zu dem der FUNKSCHAU-Redaktion vorliegenden Buch deutschsprachig ist und auch über die Sprachbarriere hinaus etwas umgearbeitet wurde. Warten wir's also ab.

Die Literatur wird auch getrennt von der Mikrocomputer-Platine einzeln erhältlich sein. Sie beinhaltet ausführliche Beschreibungen des Systems, der CPU-Befehle und der übrigen Bausteine auf der Platine. Einige kleine Beispielprogramme sind zum besseren Verständnis eingestreut. Leider konnte sich auch Texas Instruments bisher nicht entschließen, dem System eine Demonstrationskassette beizupacken, mit der der Computer-Neuling kurz nach dem Auspacken schon ein kleines Erfolgserlebnis genießen könnte.

Trotzdem: Zweifellos ist der Mikrocomputer TM 990/189 heute eine der preiswertesten Platinen mit minicomputer-ähnlichem Befehlssatz und ist, solange man nicht an umfangreiche Systemerweiterungen denkt, für Ausbildungszwecke bestens geeignet. Fe.

Warum sind manche Programme in der FUNKSCHAU nur hexadezimal aufgelistet?

Ganz einfach: Um Platz zu sparen! Um Ihnen dennoch das Verständnis für solche Zahlengruppen zu erleichtern, erschien bereits in Heft 21/1978 ein Disassembler für den Mikrocomputer KIM-1; für den 8080 folgt in Kürze ähnliches. Der AIM-65 besitzt einen solchen Disassembler bereits in seinem ROM-Monitorprogramm. Er druckt die Befehle in mnemonischer Form aus. Was das heißt? Siehe FUNKSCHAU 1979, Heft 3, Seite 153! Übrigens ist es ziemlich optimistisch, zu glauben, ein disassembliertes 6502-Programm ließe sich z. B. ohne weiteres für den 8080 umschreiben.

In den meisten Fällen läuft das auf ein völliges Neuprogrammieren hinaus.

Rudolf Hofer

Dateiverarbeitung mit dem TRS-80

Will man mit Hilfe eines Heimcomputers Dateien aufbauen, die laufend ergänzt und aktualisiert werden sollen und auf die man schnell zugreifen will, dann ist ein Floppy-Disk-Laufwerk ein unerläßliches Hilfsmittel. Der Fortschritt gegenüber dem Kassettenrecorder liegt nicht nur in der Datenübertragungsgeschwindigkeit, sondern vielmehr in der Möglichkeit, auf bestimmte Datenabschnitte (Records) gezielt zuzugreifen. Die folgenden Angaben gelten speziell für den TRS-80, sind aber grundsätzlich auch auf andere Heimcomputer übertragbar.

Sequentielle Dateien

Der TRS-80 bietet zwei grundsätzliche Möglichkeiten, Daten auf eine Diskette zu schreiben: als sequentielle Datei (*sequential file*) und als wahlfreie Datei (*random file*).

Die sequentielle Dateiverarbeitung unterscheidet sich kaum von der Datenspeicherung auf Kassette. Um Daten auf Diskette zu schreiben, benutzt man den Befehl PRINT # gefolgt von einer Datei-Nummer (File-Nr.) und den abzuspeichernden Variablen oder Ausdrükken, die jeweils durch Komma bzw. Strichpunkt getrennt sind. Ein Beispiel:

PRINT # 1, A; B; C$

speichert die numerischen Variablen A und B sowie die String-Variable C$ in der Datei Nummer 1 ab.

Werden zwei String-Variable hintereinander abgespeichert, dann sind sie durch ein zusätzliches Komma (in Anführungszeichen) zu trennen:

PRINT #1, A$; ", "; B$

Fehlt das Komma, und werden die beiden Variablen nur durch Strichpunkt getrennt, dann werden sie beim Einlesen zusammengezogen und als eine einzige interpretiert.

Eingelesen wird eine sequentielle Datei mit dem INPUT-Befehl. Um die Werte des ersten Beispiels den Variablen X, Y und Z$ zuzuweisen, wäre folgender Befehl nötig:

INPUT # 1, X, Y, Z$

Die Datei-Nummer, in diesem Fall 1, muß natürlich der im PRINT-Befehl entsprechen; sie darf eine Ganzzahl von 1...16 sein.

Vor dem Einschreiben und Auslesen muß die entsprechende Datei geöffnet und geschlossen werden. Das geschieht mit den Befehlen OPEN und CLOSE. Sie haben folgendes Format:

OPEN "Art", #File-Nr., "File-Name"

und

CLOSE File-Nr., File-Nr., ...

Entfällt bei CLOSE jeglicher Zusatz, werden alle Dateien geschlossen (auch bei den nachfolgend besprochenen wahlfreien Dateien). Beim OPEN-Befehl sind drei verschiedene *Arten* möglich. Sie werden durch die Buchstaben O, I und R ausgedrückt. Mit "O" wird eine sequentielle Datei eröffnet, in die nachfolgend geschrieben werden soll (*Output*); "I" wird verwendet, wenn man lesen will (*Input*). Wahlfreie Dateien werden in jedem Fall mit OPEN "R", # File-Nr., "File-Name" eröffnet – ganz gleich, ob man schreiben oder lesen will.

Während im Level-II-BASIC innerhalb von String-Variablen bestimmte Zeichen (z. B. Anführungszeichen und Kommas) nicht vorkommen dürfen, gilt diese Beschränkung im erweiterten Disk-BASIC nicht mehr: Mit dem Befehl LINE INPUT, der wie der normale INPUT-Befehl benutzt wird, weist man einer Variablen einen beliebigen Text zu. Er kann auch im Zusammenhang mit sequentiellen Dateien verwendet werden.

Um festzustellen, ob alle Daten gelesen wurden, ist die Funktion EOF vorhanden. Sie hat folgendes Format:

EOF (File-Nr.)

Sind alle Daten gelesen, liefert sie den Wert –1, andernfalls 0. Jede sequentielle Datei benötigt auf der Diskette mindestens 1280 Byte. Bei Bedarf werden jeweils weitere 1280 Byte belegt.

Wahlfreie Dateien

Wahlfreie Dateien bringen die eingangs erwähnte Möglichkeit, auf bestimmte Dateien gezielt zuzugreifen. Im Gegensatz zu sequentiellen Dateien speichern sie die Daten nicht im ASCII-Format, sondern in komprimierter Form. Außerdem kann nach dem OPEN-Befehl beliebig geschrieben oder gelesen werden.

Grundelement für jede Schreib/Lese-Operation ist ein "Record" (256 Byte). Jede Datei kann davon bis zu 329 enthalten. Dementsprechend wird jedem Record eine Nummer von 1...329 zugeordnet. Bevor ein Record auf die Diskette geschrieben wird, werden die Daten in einem Puffer abgelegt (obwohl der Puffer 256 Byte lang ist, dürfen nur 255 Byte benutzt werden). Mit dem FIELD-Befehl weist man verschiedenen Variablen ihren Platz im Puffer zu. Analog dazu liest man einen Record von der Diskette zuerst in den Puffer und entnimmt dann die Variablen-Werte aus den vorher vereinbarten Plätzen. Dazu ist der gleiche FIELD-Befehl nötig. Ein Beispiel soll das Verfahren verdeutlichen: In die Datei Nummer 2 mit dem Namen ADRESS sollen zunächst die String-Variablen VN$ (= Vorname, max. 10 Zeichen), NN$ (= Nachnahme, max. 15 Zeichen) und OR$ (= Ort, max. 15 Zeichen) eingeschrieben werden. Die Record-Nummer ist 65. Folgendes Programm erfüllt die Aufgabe:

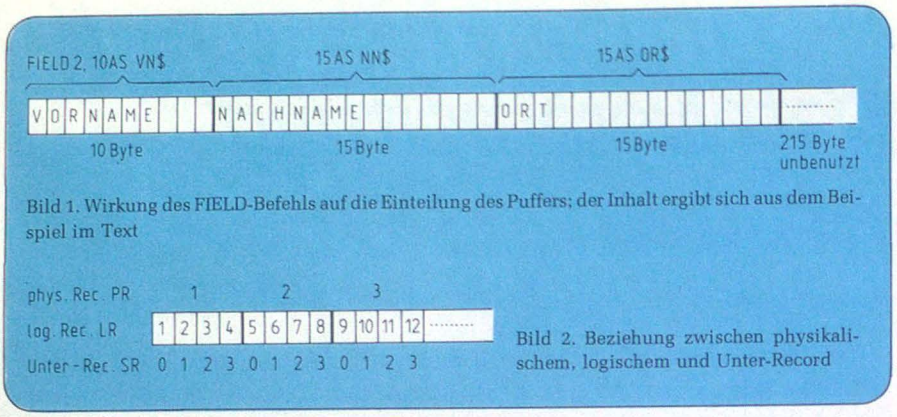


Bild 1. Wirkung des FIELD-Befehls auf die Einteilung des Puffers; der Inhalt ergibt sich aus dem Beispiel im Text

Bild 2. Beziehung zwischen physikalischem, logischem und Unter-Record

```
100 OPEN "R", 2, "ADRESS"
110 FIELD 2, 10 AS VN$, 15 AS NN$, 15
AS OR$
120 LSET VN$ = "VORNAME" : LSET
NN$ = "NACHNAME" : LSET OR$ =
"ORT"
130 PUT 2,65
```

Zeile 110 bewirkt, daß den Variablen der Reihe nach 10, 15 und noch einmal 15 Byte zur Verfügung gestellt werden. In Zeile 120 werden sie mit LSET linksbündig in den Puffer übernommen (RSET = rechtsbündig). Der Einfachheit halber wurde ihnen hier direkt ein String zugewiesen (*Bild 1*). Natürlich kann ihnen auch der Wert einer anderen Variablen zugewiesen werden, der z. B. vorher mit Hilfe von INPUT oder LINE INPUT eingetippt wurde. An dieser Stelle ist zu vermerken, daß alle Variablen als Strings abgespeichert werden. Numerische Variablen muß man deshalb vorher konvertieren. Für die drei möglichen Typen "Integer", "Single Precision" und "Double Precision" sind dazu die Funktionen MKI$, MKS$ und MKD$ vorgesehen. Im Puffer müssen für sie 2, 4 und 8 Byte reserviert werden. Wollte man also im vorhergehenden Beispiel die Variablen A%, B und C# abspeichern, müßte man die Zeilen 110 und 120 ersetzen durch:

```
110 FIELD 2, 2 AS A$, 4 AS B$, 8 AS C$
120 LSET A$ = MKI$(A%) : LSET B$ =
MKS$(B) : LSET C$ = MKD$ (C#)
```

Zeile 130 schließlich bringt den Pufferinhalt auf die Diskette. Wird die Record-Nummer im PUT-Befehl nicht angegeben, nimmt der Computer Eins an und inkrementiert dann bei jeder weiteren Schreiboperation.

Um einen Record zu lesen, benutzt man dieselbe FIELD-Zeile wie beim Schreiben. Danach holt man ihn mit GET File-Nr., Record-Nr. in den Puffer. Die Variablen haben jetzt die abgespeicherten Werte. Numerische Variablen werden mit CVI, CVS und CVD in ihr ursprüngliches Format konvertiert.
Beispiel: A% = CVI (A$)
Der im ersten Beispiel abgespeicherte Record kann mit diesem Programm gelesen werden:

```
200 OPEN"R", 2, "ADRESS"
210 FIELD 2, 10 AS VN$, 15 AS NN$, 15
AS OR$
220 GET 2, 65
```

Innerhalb einer Datei ist es zulässig, verschiedenen Records verschiedene Feldformate zuzuweisen. Auch beim Lesen muß nicht unbedingt dasselbe Format gewählt werden wie beim Schreiben. Es ist z. B. denkbar, die getrennt abgespeicherten Vor- und Nachnamen als eine Variable wieder von der Diskette zu holen.

Will man zu einer Datei Records lükkenlos anfügen, leistet die LOF-Funktion wertvolle Dienste: LOF (File-Nr.) liefert die Nummer des letzten Records.

Was tun gegen Platzverschwendung?

In vielen Fällen wird man abzuspeichernde Daten in kleinere Portionen als 255 Byte einteilen. Wertvoller Speicherplatz bleibt dann ungenutzt. Um dies zu verhindern, empfiehlt das Radio-Shack-Handbuch [1] ein Verfahren, das folgendermaßen aussieht:

Die physikalischen Records werden vom Programm in je vier Unter-Records (*Subrecord*) eingeteilt, die mit Subrecord-Nummern (SR = 0...3) versehen werden. Die neu entstandenen logischen Records beinhalten maximal 63 Byte; sie werden mit logischen Record-Nummern (LR) identifiziert, die mit 1 beginnen. *Bild 2* zeigt das Schema. Bei bekannten LR können PR und SR berechnet werden:

a) PR = INT ((LR-1)/4)+1
b) SR = LR-4* INT ((LR-1)/4)–1

Diese beiden Werte sind nötig, um genau die Portion in den oder aus dem Puffer zu bringen, die dem logischen Record entspricht. Nehmen wir an, N$ sei die interessierende Variable; D$ ist eine Hilfs-Variable. Mit

```
FIELD 3, SR* 63 AS D$, 63 AS N$
```

wird N$ einem von vier Unter-Record-Bereichen zugewiesen. Der physikalische Record, der nach wie vor im PUT- und im GET-Befehl eingesetzt wird, kann wie angegeben errechnet werden. Natürlich ist es auch möglich, N$ durch mehrere andere Variable zu ersetzen. Zusammen dürfen sie 63 Byte belegen – sie müssen es aber nicht. Ebenso steht es einem frei, nicht vier, sondern eine andere Zahl von Unter-Records zu bilden. Eine intensive Verwendung dieser Unter-Record-Technik demonstriert ein „Datenbank"-Programm, das demnächst in der FUNKSCHAU erscheinen wird.

Literatur

[1] Disk Basic Version 1.1, TRS DOS Version 2.2. Handbuch der Fa. Radio Shack, Juli 1978.

bit/s oder Baud?

Die Übertragungsgeschwindigkeit von Modems und Leitungen zum seriellen Datentransfer wird manchmal in bit/s, manchmal aber auch in Baud angegeben. Beide Einheiten sind nicht miteinander gleichzusetzen: bit/s bezeichnet die tatsächlich in einer Sekunde übertragene Datenmenge, wobei auch Synchron- und Prüfbits mitzählen.

Mit „Baud" bezeichnet man dagegen den Kehrwert der kürzestmöglichen Impulsdauer bei der Datenübertragung. Normalerweise ist diese mit einer Bitlänge identisch; bei mehrwertigen Übertragungsverfahren ist dies jedoch nicht mehr der Fall: Codiert man z. B. ein Bit als Frequenzumtastung und ein zweites *gleichzeitig* als Amplitudensprung, so wird – grob gesagt – eine Übertragung von z. B. 2400 bit/s bei nur 1200 Baud möglich, wobei die Bandbreite der Leitung für die Baudrate, nicht für die Bitrate dimensioniert werden muß. Fe.

KIM decodiert Morsezeichen

Für das Erlernen von Morsezeichen hat mancher Funkamateur schon kostbare Monate seines Lebens verwendet, und auch mit einiger Übung schafft man erst, etwa 60...100 Buchstaben pro Minute aufzunehmen. Mit Mikrocomputern lassen sich Morsezeichen dagegen bequem auch noch z. B. bei 300 BpM aufnehmen. Das vorliegende Programm stellt sie auf dem Display des KIM-1 als durchlaufende Schrift dar.

Hardware-Voraussetzungen

Zum Empfang von Morsezeichen benötigt man selbstverständlich erst einmal einen Empfänger, z. B. für die Kurzwellen-Amateurfunkbereiche, an dessen Ausgang eine Schaltung angeschlossen wird, die die empfangenen Morsezeichen in ein verwertbares Digitalsignal umformt. Eine solche Schaltung befindet sich in FUNKSCHAU 1979, Heft 9, Seite 530, unter dem Titel „Ein einfacher Fernschreibkonverter". Sie verwendet eine PLL-Schaltung (NE 567) mit „Open-Collector"-Ausgang. Pin 8 des IC wird nun einfach mit dem KIM-Eingangsport PB 0 verbunden. Das ist alles, was man außer dem KIM selbst zum Empfang der Morsezeichen benötigt. Unter Umständen lassen sich mit aufwendigeren Filterschaltungen zwar bessere Empfangsergebnisse erzielen, doch genügt die Störsicherheit einer PLL-Schaltung in den meisten Fällen vollkommen. Als Kontrollanzeige sollte man die Serienschaltung einer LED und eines Widerstandes von 1 kΩ von Pin 8 des IC nach +5 V legen.

Nichts ist perfekt

Es wäre übertrieben, zu sagen, daß ein Mikrocomputer das Erlernen von Morsezeichen überflüssig macht. Wer das hier vorgestellte Programm ausprobiert, wird bald merken, daß Mikrocomputer sehr schnell ihre Grenzen finden; auch Signale, die man „mit dem Ohr" noch gut mithören kann, können bereits zu Schwierigkeiten führen, besonders, wenn die sendende Station eine recht unsaubere Gebeweise hat und dadurch das Zeitschema des Mikroprozessors durcheinanderbringt. Dagegen gibt es selten Schwierigkeiten, wenn auch die Gegenstation einen Mikrocomputer zum Senden verwendet (vgl. FUNKSCHAU 1979, Heft 14, KIM als Morseschreibmaschine) oder mit einer elektronischen, automatischen oder halbautomatischen Taste sendet.

Bezieht man das „Timing" der Morsezeichen auf die Dauer eines Punktes, so ist der Zwischenraum zwischen den Elementen eines Buchstabens ebenfalls einen Punkt lang; ein Strich besitzt drei Punktlängen, und der Zwischenraum zwischen zwei Buchstaben ist auch drei Punkte lang. Zwischen zwei Worten macht man üblicherweise mindestens fünf Punktlängen Pause.

So funktioniert das Decodieren

Das *Bild* zeigt das hexadezimal aufgelistete Programm für den KIM-1. Die Startadresse ist 0000. Wer statt PB 0 einen anderen Port als Eingang verwenden möchte, kann die entsprechende Maske für PB an den Adressen 000E, 0050 und 0066 ändern.

Zum Decodieren mißt das Programm zunächst die Länge des empfangenen Zeichenelementes (Adresse 0004). Als „Timer" wird dabei die Verzögerung einer alphanumerischen Anzeigeroutine (00A0) verwendet, die weitgehend der Routine aus dem Hobbycomputer-Sonderheft des Franzis-Verlages (S. 38) entspricht. Ist das Zeichen kürzer als zwei Punktlängen, so wird es als Punkt, andernfalls als Strich interpretiert. Wenn auf dem KIM-Tastenfeld die Taste 0 gedrückt wird, so schaltet das Programm auf automatische Geschwindigkeitsanpassung um: Wurde das Zeichenelement als Punkt erkannt, so wird die gerade gemessene Zeit als neue Punktlänge übernommen. Störimpulse, die kürzer als eine halbe Punktlänge sind, werden allerdings ignoriert (006C); ihre Länge wird zur Pausezeit addiert. Dies hat jedoch zur Folge, daß die Automatik nicht über den ganzen

Hexadezimale Auflistung des Morse-Decodierprogramms für den KIM-1. Die Startadresse ist 0000

```
0000  A9 01 85 EE A9 FF 85 ED 20 A0 00 E6 ED A9 01 2C
0010  02 17 F0 F4 A5 ED C5 EC 90 EA C5 EB B0 1E 06 EE
0020  A6 E6 D0 1C A6 ED A4 ED CA 88 88 10 FB 86 EC 18
0030  A5 ED 65 ED 85 EB 65 EB 85 EA D0 04 06 EE E6 EE
0040  A9 FF 85 E8 20 A0 00 E6 E8 A5 EA C5 E8 90 34 A9
0050  01 2C 02 17 D0 EE A9 FF 85 E7 20 A0 00 E6 E7 A5
0060  E7 C5 EC B0 11 A9 01 2C 02 17 F0 EE A5 E7 18 65
0070  E8 85 E8 4C 44 00 A5 E7 85 ED A5 EB C5 E8 90 13
0080  4C 08 00 20 1D 01 20 27 01 20 1D 01 A9 00 85 E0
0090  4C 00 00 20 1D 01 20 27 01 A9 01 85 EE 4C 0D 00
00A0  A2 03 86 E9 A9 7F 8D 41 17 A2 09 A0 06 B9 DF 00
00B0  84 DF A0 00 8C 40 17 8E 42 17 8D 40 17 A0 20 88
00C0  D0 FD E8 E8 A4 DF 88 D0 E4 8E 42 17 A9 00 8D 41
00D0  17 A6 E9 CA D0 CC 20 40 1F 4C 00 01 00 00 00 05

0100  20 6A 1F C9 15 F0 15 85 E6 C9 00 F0 0F 85 EC 18
0110  65 EC 85 EB 65 EB 85 EB 65 EB 85 EA 60 A2 04 B5
0120  E0 95 E1 CA 10 F9 60 A5 EE A2 FF E8 E0 30 F0 0B
0130  DD 40 01 D0 F6 BD 70 01 85 E0 60 A9 08 85 E0 60
0140  05 18 1A 0C 02 12 0E 10 04 17 0D 14 07 06 0F 16
0150  1D 0A 08 03 09 11 0B 19 1B 1C 2F 27 23 21 20 30
0160  38 3C 3E 3F 4C 31 61 32 55 5E 52 36 6D 2A 45 22
0170  77 7C 39 5E 79 71 3D 74 04 1F 78 38 37 54 5C 73
0180  E7 50 6D 31 3E 1C 7E 76 6E 5B 06 5B 4F 66 6D 7D
0190  07 7F 6F 3F 53 40 40 52 0C 02 22 61 43 41 49 1C
```

Zero-Page-Adressen

EE Zeichenpuffer	E9 Zeitbasis-Zähler
ED Zähler für die Elementlänge	E8 Zähler f. Pausenlänge
EC Halbe Punktlänge	E7 Störimpuls-Zähler
EB Zwei Punktlängen	E6 Automatik-Flag
EA Vier Punktlängen	DF Save Y (f. Display)

Der angezeigte Text steht im Adressenbereich 00E0...00E5.

Variationsbereich arbeitet. Vor dem Einschalten der Automatik muß deshalb das Empfangstempo mit einer der manuellen Geschwindigkeitswahltasten auf den Fangbereich der Tempoautomatik vorgesetzt werden. Um diesen Nachteil zu vermeiden, müßte auf eine Störimpulsunterdrückung verzichtet werden, was aber noch schlechtere Ergebnisse bringen würde. Bei einer Pause von mehr als zwei Punktlängen wurde ein vollständiges Zeichen empfangen und über eine Umwandlungstabelle für die Morsezeichen (0140...016F) und das Siebensegment-Alphabet (0170...019F) in das KIM-Display geschoben. Dabei wird ein ASCII-ähnlicher Zwischencode verwendet.

Tritt eine Pause von mehr als vier Punktlängen auf, so wird sie als Wortzwischenraum interpretiert, und in das Display wird zusätzlich eine Leerstelle geschoben.

Manuelle und automatische Geschwindigkeitswahl

Wie schon erwähnt, kann sich der KIM-1 automatisch an die Geschwindigkeit der empfangenen Morsezeichen anpassen. Verständlicherweise funktioniert das aber nur in gewissen Grenzen und bei relativ wenig gestörten Signalen. Aus diesem Grunde steht zusätzlich eine manuelle Geschwindigkeitswahl zur Verfügung, die ebenfalls über die KIM-Tastatur erfolgt. Die Taste 1 ergibt das schnellste Tempo, nach F hin wird es langsamer. Beim Drücken der Taste 0 wird die Automatik eingeschaltet.

Einen Anhaltspunkt für die richtige Geschwindigkeit bilden die in der Anzeige erscheinenden Zwischenräume. Erscheint nach jedem Buchstaben ein Zwischenraum, so läuft der KIM zu schnell; läuft er dagegen zu langsam, so „kleben" auch die Worte aneinander. Im praktischen Betrieb hat sich die manuelle Geschwindigkeitswahl als wesentlich günstiger erwiesen. Erstens schwanken die Geschwindigkeiten der empfangenen Stationen meist nicht allzu sehr, und zweitens kann die Automatik bei anhaltenden Störungen doch außer Tritt geraten.

Noch ein Tip: Speichert man das Programm auf Band, so ist es ganz nützlich, wenn man ab der Zelle 00E0 die Daten 53 5C 73 37 79 31 einschreibt. Beim Programmstart erscheint dann im Display zunächst die Frage „Tempo?", um auf die Notwendigkeit der Geschwindigkeitswahl hinzuweisen, da nach jedem Programmstart der KIM vorerst auf einem sehr langsamen Tempo steht.

Gerhard Kirchberger
Herwig Feichtinger

FSK-Demodulation per Software

Ein Modem besteht aus Modulator und Demodulator zur Übertragung von Daten auf einer Tonfrequenzleitung, z. B. per Telefon oder Funk. Meist verwendet man dazu ein Zwei-Frequenzen-Verfahren, auf Neuhochdeutsch Frequency Shift Keying (FSK). In FUNKSCHAU 14 und 15/1978 wurde ein solches Modem unter der Rubrik „Schaltungskniffe" beschrieben. Natürlich läßt sich die dafür erforderliche Hardware durch Software reduzieren.

Hier wird nun ein Demodulator beschrieben, der aus einem FSK-Signal (z. B. Kansas-City-Standard) ein Digitalsignal rückgewinnt, das entweder einem Terminal zugeleitet oder aber vom Mikrocomputer selbst weiterverarbeitet werden kann.

Der Demodulator arbeitet nach einem einfachen Algorithmus, der aus *Bild 1* hervorgeht. Nach dem Programmstart wartet er auf eine (positive oder negative) Flanke des Eingangssignales. Nach einer Verzögerungszeit von 3/4 der Periodenzeit der höheren Frequenz wird der Eingang erneut abgefragt. Hat sich sein Zustand geändert, so handelt es sich um die tiefere Frequenz und damit um log. 0; andernfalls liegt die höhere Frequenz (log. 1) an.

Das einfache Unterprogramm *(Bild 2)* für den Mikroprozessor 6502 geht davon aus, daß sich außer dem Nf-Eingang kein I/O-Port ändert. Die Adressenbelegung entspricht dem Mikrocomputer KIM-1, wobei der TTL-Nf-Eingang ein beliebiger Pin des Ports PB sein kann. Bei der Rückkehr aus dem Unterprogramm ist bei der höheren Frequenz das Zero-Flag gesetzt, bei der tieferen rückgesetzt, was leicht ausgewertet werden kann.

Die Abfrage nach 3/4 der Periodendauer der höheren Frequenz ist nur dann optimal, wenn sich die beiden FSK-Frequenzen etwa 1:2 verhalten. Andernfalls ist der Abfragezeitpunkt geringfügig zu ändern.

Übrigens hat die Software-Methode auch einen kleinen Nachteil etwa gegenüber Phase-Locked-Loop-Demodulatoren: Der erforderliche Störabstand muß mindestens etwa 6 dB betragen, während eine PLL noch mit 0...3 dB auskommt. Bei Tonband-Aufzeichnungen spielt dies überhaupt keine Rolle, da schon bei einfachen Kassettenrecordern der Störabstand über 30 dB liegt; bei Funk- oder Fernsprech-Übertragungen können sich jedoch manchmal Probleme ergeben. Hier ist es sinnvoll, selektive Filter und Begrenzerverstärker einzusetzen, um die Störbandbreite auf das für die Übertragung wirklich notwendige Frequenzband zu verringern und Amplitudenschwankungen zu beseitigen. Und während es wegen der unvermeidlichen Gleichlaufschwankungen bei Kassettengeräten meist besser ist, die Filterbandbreite nicht zu schmal zu machen, wird man bei Funkübertragungen die Bandbreite möglichst der Baudrate anpassen. Fe.

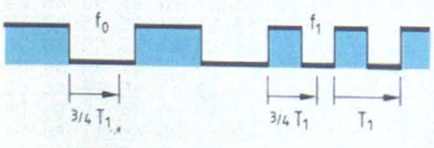


Bild 1. Fragt man ein Rechtecksignal zum richtigen Zeitpunkt ab, so läßt sich leicht feststellen, ob die höhere oder tiefere Frequenz des FSK-Signals anliegt

```
0200 AD 02 17 LDA 1702
0203 CD 02 17 CMP 1702
0206 F0 FB    BEQ 0203
0208 A2 3B    LDX # 3B
020A CA       DEX
020B D0 FD    BNE 020A
020D CD 02 17 CMP 1702
0210 60       RTS
```

Bild 2. Einfaches Unterprogramm zur FSK-Decodierung mit dem 6502. Liegt die höhere der beiden Frequenzen an, so ist bei der Rückkehr aus dem Unterprogramm im Status-Register das Zero-Flag gesetzt. Die Dimensionierung erfolgte für 1275/2125 Hz bei 1 MHz CPU-Taktfrequenz

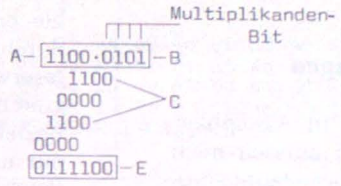
Hans Werner

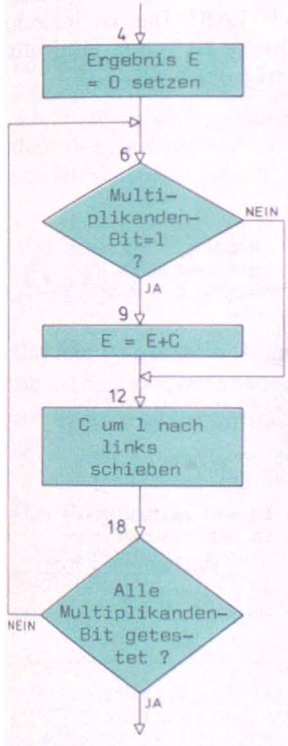
Mikrocomputer

Eine kleine Einführung

22. Teil

Sehen wir uns einmal das Flußdiagramm für die Aufgabe A·B = E an. C entspricht dem Multiplikator A, allerdings um 1,2... Stellen verschoben.





Speicherzelle für das Ergebnis reservieren und den Inhalt auf 0 setzen.

In der Reihenfolge von rechts nach links jedes Bit von B testen.

Ist es 1: C addieren (E+C)
Ist es 0: keine Addition

C für die nächste Addition vorbereiten, d. h. um eine Stelle nach links schieben.

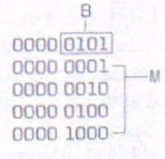
Nach vier Durchläufen sind alle Bits getestet und die erforderlichen Additionen durchgeführt. In der Ergebniszelle steht das Produkt E = A·B.

```
 0   LA AC d  A      A in den Akku laden
     SP AC d  C      A in Zelle C speichern
     LA AC k  1      Zahl 1 in den Akku laden
     SP AC d  M      1 in M speichern = Vergleich-
                                        muster
 4   LA AC k  0      Zahl 0 laden
     SP AC d  E      Ergebniszelle E gleich 0 setzen

 6   LA AC d  B      B in den Akku laden
     UN    d  M      B UND M
     BR EO d 12      Ergebnis = 0: Nach 12 springen

 9   LA AC d  E      E in den Akku laden
     AD    d  C      E + C
     SP AC d  E      Ergebnis E in E speichern

12   LA AC d  C      C in den Akku laden
     SH LI    1      C um eine Stelle nach links
     SP AC d  C      Neues Muster in C speichern

15   LA AC d  M      Vergleichmuster M laden
     SH LI    1      M um eine Stelle nach links
     SP AC d  M      Neues Muster in M speichern

18   UN    k 16      M UND 16 (10000)
     BR EO d  6      Wenn M  10000 erreicht hat,
                     sind alle Bit's getestet und es
20   END             erfolgt kein Rücksprung nach 6.
```

ACHTUNG !
Die Programmarken 0,4,6... geben nicht die Speicherplatz-nummer an, denn die Befehle sind länger als 1 Byte.

Das Programm

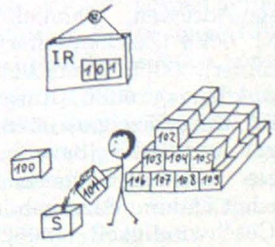
Das Multiplikanden-Bit wird mit Hilfe eines Vergleichsmusters M und der UND-Verknüpfung B UND M getestet. Beim ersten Durchlauf ist M = 1, beim zweiten Durchlauf 2 usw.



Die symbolischen Adreßbezeichnungen sind gleichzeitig die Namen der entsprechenden Variablen A, B usw. Auf eine Variablenliste wurde verzichtet. Neu ist der Sprungbefehl BR EO n: Springe nach n, wenn das Ergebnis der Operation gleich 0 ist.

Programm mit indizierten Befehlen

Mit indizierten Befehlen können sehr leicht ganze Speicherbereiche angesprochen werden. Als Beispiel soll der Inhalt der Speicherzellen 100 bis 300 addiert werden:



Summe S = (100) + (101) + ... (300)

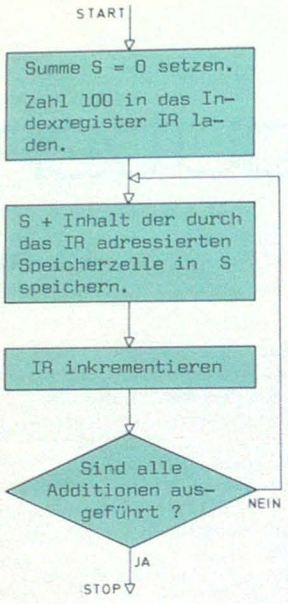
Drei neue Befehle ermöglichen das Programm:

LA IR: Lade das Indexregister.

IN IR: Inkrementiere das Indexregister.

BR EX: Springe, wenn das Ergebnis einer Operation ungleich Null ist.

Die Distanzadresse des indizierten Additionsbefehls ist 0, d. h. das Indexregister gibt alleine die Adresse an.




```
1. Durchlauf: S+(100)= S
2.          : S+(101)= S
201.        : S+(300)= S
```

Frage:
In den letzten beiden Programmen sind die unmittelbaren Befehle nachlässig geschrieben. Was wurde vergessen?

Antwort:
Hinter den Konstanten fehlt die Angabe D, H bzw. B. In allen Fällen waren dezimale Zahlen gemeint. Richtig ist: LA IR k 100D usw.

Assembler-Anweisungen

Die Programme sind in Assemblersprache formuliert. Sie müssen noch durch ein spezielles Übersetzungsprogramm in den Maschinencode „assembliert" werden. Erst dann ist für den Mikrocomputer ein ablauffähiges Programm vorhanden. Dazu erfinden wir noch zwei Anweisungen, die ausschließlich an das Übersetzungsprogramm gerichtet sind.

ORG n

Das Programm wird ab Zelle n im Speicher abgelegt (organisiert).

VARI m

Jede symbolische Adresse bzw. Variable benötigt einen Speicherplatz, der durch die Anweisung VARI in Zelle m reserviert wird. VARI erhält als Programmmarke die symbolische Adresse. Beziehen sich symbolische Adressen nur auf die Programmarken (z. B. M bei Befehl BR EX d), entfällt die Platzreservierung durch VARI. Das ist logisch, denn die Adresse ist schon eindeutig durch die Marke definiert.

```
      LA AC k   0     Speicherzelle für die Summe S
      SP AC d   S     reservieren und 0 setzen.
      LA IR k   100   Zahl 100 in das IR laden
      LA AC k   201   Anzahl der erforderlichen
      SP AC d   Z     Schritte in Z speichern.
M     LA AC d   S     S in den Akku laden
      AD    i   0     ((IR)+0) + (AC) in den Akku
      SP AC d   S     Summe in Zelle S speichern
      IN IR           Inhalt IR inkrementieren (+1)
      DE    d   Z     Inhalt Z dekrementieren (-1)
      LA AC d   Z     Solange Z ungleich 0 ist,
      BR EX d   M     Sprung nach M
      END
```

Vollständiges Programm

```
      ORG       0D    — Das Programm steht ab Speicherzelle 0 im ROM
      LA AC k   0D
      SP AC d   S
M     LA AC d   S     — M bedeutet an dieser Stelle: Programmarke (auf eng. LABEL)
      BR EX d   M     — M bedeutet an dieser Stelle: symbolische Adresse / Variable
S     VARI      300D  — Platz 300 im RAM reservieren
Z     VARI      301D  — Platz 301 im RAM reservieren
      END
```

Fortsetzung folgt

AIM-65 als Morse-Schreibmaschine

In Heft 14/1979 war beschrieben, wie sich der Mikrocomputer KIM-1 zusammen mit einem ASCII-Terminal zum Erzeugen von Morsezeichen einsetzen läßt. Der AIM-65 (und damit auch der PC-100 von Siemens) besitzt bereits eine ASCII-Tastatur, arbeitet jedoch mit anderen Timer- und Monitorprogramm-Funktionen, so daß ein Umschreiben des KIM-Programmes auf ihn kaum möglich war, obwohl in beiden Fällen die gleiche CPU benutzt wird.

Während das KIM-Programm die ASCII-Zeichen in einer Interrupt-Routine von der Tastatur holte und im „Hauptprogramm" als Ton aussandte, arbeitet das AIM-Programm (*Bild 1*) genau umgekehrt: Das Senden der Morsezeichen geschieht per Interrupt-Routine; das Hauptprogramm (Startadresse 0200) holt die ASCII-Zeichen von der Tastatur und speichert sie in einem Ausgabebuffer (FIFO) ab, der im Adressenbereich 0300...03FF liegt. Der Interrupt wird periodisch vom Timer 1 im 6522-VIA ausgelöst. Der Timer arbeitet im freilaufenden Modus und erzeugt am Port PB 7 gleichzeitig die Tonfrequenz.

Das Programm belegt eine Reihe von Zero-Page-Adressen, nämlich 00E2 (Save X), 00E3 (Zeichenbuffer), 00E4 (Punktzähler), 00E5 (Zeitschleife für eine Punktlänge), 00E6 (Ausgabezeiger), 00E7 (Eingabezeiger), 00E8...00FC (Anzeige) und 00FF (Save X). Alle wichtigen Betriebsparameter lassen sich gezielt ändern; dazu gehören die Morse-Geschwindigkeit (0062), die Pause zwischen den einzelnen Zeichen (0085) und Worten (009C) sowie die erzeugte Tonfrequenz (0215, 021A). Sie beeinflußt auch die Geschwindigkeit.

Drückt man die Taste „Return", so erfolgt ein sofortiger Sprung zum AIM-Monitorprogramm (022E), ohne die laufende Ausgabe zu unterbrechen. Die Taste F2 führt ebenfalls zu einem Sprung in das Monitorprogramm (00B3), allerdings erst, wenn das letzte Morsezeichen ausgegeben ist, d.h. sobald Eingabe- und Ausgabezeiger übereinstimmen. Dank des Ausgabebuffers darf man bis zu 255 Zeichen vorausschreiben; das Format der Morsezeichen im Buffer stimmt mit dem des erwähnten KIM-Programmes überein, d.h. die einzelnen Bits werden, beim

```
0000 80 56 80 80 80 80 80 80 80 80 80 80 CE 8C 56 94
0010 FC 7C 3C 1C 0C 04 84 C4 E4 F4 56 56 80 80 80 32
0020 4C 60 88 A8 90 40 28 D0 08 20 78 B0 48 E0 A0 F0
0030 68 D8 50 10 C0 30 18 70 98 B8 C8 80 80 00 80 80

0040 85 FC A2 00 86 FF B5 E9 95 E8 09 80 20 7B EF A6
0050 FF E8 E0 14 D0 EE A5 FB 60 00 00 00 48 C6 E5 D0
0060 42 A9 A0 85 E5 C6 E4 F0 14 10 38 06 E3 F0 15 B0
0070 04 A9 01 D0 02 A9 03 85 E4 A9 C0 D0 02 A9 40 8D

0080 0B A0 D0 1F A9 02 85 E4 86 E2 A6 E6 E4 E7 F0 11
0090 BD 00 03 F0 15 85 E3 C9 80 D0 04 A9 08 85 E4 E6
00A0 E6 A6 E2 2C 04 A0 68 40 A2 FF 9A A9 00 8D 0B A0
00B0 8D 0E A0 6C FC FF

0200 A9 5C 8D 04 A4 A9 00 8D 05 A4 A9 40 8D 0B A0 A9
0210 C0 8D 0E A0 A9 00 8D 04 A0 A9 02 8D 05 A0 A9 00
0220 A2 16 95 E6 CA 10 FB 20 40 EC C9 0D D0 03 6C FC
0230 FF 20 40 00 AA BD E0 FF A6 E7 E6 E7 9D 00 03 18
0240 90 E5        AIM-65-MORSESCHREIBMASCHINE
```

◀ Bild 1. Programm zum Erzeugen von Morsezeichen mit den Mikrocomputern AIM-65 von Rockwell oder PC-100 von Siemens. Startadresse ist 0200, Nf-Ausgang ist der Port PB 7

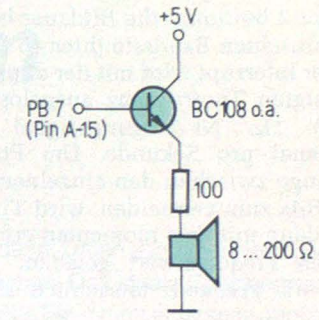


Bild 2. Anschluß eines Lautsprechers an den Mikrocomputer

höchstwertigen beginnend, als Punkte (0) und Striche (1) interpretiert, wobei die letzte 1 nicht gesendet wird, sondern das Zeichenende anzeigt. Selbstverständlich können in der Umwandlungstabelle (0000....003F), die die ASCII-Zeichen hex 20...5F umfaßt, auch Sonderzeichen programmiert werden. *Bild 2* zeigt den Anschluß eines kleinen Lautsprechers an den AIM-65. Fe.

AIM-65 sendet RTTY

Das hier vorgestellte Programm für den Mikrocomputer AIM-65 enthält eine interessante Lösung zum Erzeugen eines Frequenz-Umtast-Signales (FSK) für die Modulation eines Funkfernschreibsenders. Es benutzt zwei programmierbare Timer im VIA 6522.

Das Programm

Funkamateure führen Fernschreib-Verbindungen (RTTY) heute meist noch im 5-bit-Baudot-Code durch. Will man solche Sendungen mit dem AIM-65 empfangen, so kann man sich des in Heft 15, Seite 884, veröffentlichten Programms bedienen, das die empfangenen Zeichen als durchlaufende Schrift auf dem 20stelligen alphanumerischen Display darstellt. Bei den üblichen 45 Bd (bit/s) kann man noch gut mitlesen.

Will man den AIM-65 (oder natürlich den PC-100 von Siemens) auch zum Senden benutzen, so führt dies zu folgenden Forderungen:

1. Der ASCII-Code des Tastenfeldes muß in ein bitserielles Baudot-Signal umgewandelt werden.
2. Da man u. U. manchmal schneller als 45 Bd schreibt, muß ein Bufferspeicher (FIFO) vorhanden sein.
3. Das Programm sollte auf Tastendruck einen vorher gespeicherten Standardtext, z. B. das eigene Rufzeichen nebst Standort, senden können.

Das in *Bild 1* hexadezimal aufgelistete Programm sorgt außerdem dafür, daß der notwendige externe Hardware-Aufwand minimal bleibt; es erzeugt nämlich das Baudot-Ausgangssignal direkt als Tonfrequenz.

```
0200 A9 A1 8D 04 A4 A9 02 8D
0208 05 A4 A9 00 85 E8 85 E9
0210 20 EF EC A9 80 8D 02 A0
0218 A9 C0 8D 0B A0 8D 0E A0
0220 A9 00 8D 05 A0 58 A2 10
0228 A9 1F 20 67 02 CA D0 F8
0230 20 F0 E9 A9 0D 20 72 02
0238 20 73 E9 C9 0D F0 F1 C9
0240 5B F0 15 AE 15 A4 E0 32
0248 30 EB C9 20 F0 E2 C9 2D
0250 D0 E3 20 72 02 4C 30 02
0258 A2 00 A9 0D 20 72 02 BD
0260 70 01 F0 CC E8 D0 F5 A4
0268 E8 E6 E8 0A 09 C0 99 00
0270 03 60 A0 00 D9 68 00 F0
0278 03 C8 D0 F8 98 48 29 20
0280 C5 FD F0 0F 85 FD C9 00
0288 F0 04 A9 1B D0 02 A9 1F
0290 20 67 02 68 C9 08 D0 05
0298 20 67 02 A9 02 20 67 02
02A0 60
```

Bild 1. Das Hauptprogramm weist keine Besonderheiten auf und wurde deshalb nur hexadezimal aufgelistet. Das Unterprogramm bei 0272 wandelt ein ASCII-Zeichen in den Baudot-Code um und speichert es in den FIFO

```
02A1 2C 09 A0 BIT A009
02A4 10 48    BPL 02EE
02A6 48       PHA
02A7 A9 57    LDA #57
02A9 8D 09 A0 STA A009
02AC A5 EA    LDA EA
02AE F0 23    BEQ 02D3
02B0 38       SEC
02B1 66 FE    ROR FE
02B3 AD 07 A0 LDA A007
02B6 90 0B    BCC 02C3
02B8 F0 15    BEQ 02CF
02BA A9 E9    LDA #E9
02BC 8D 04 A0 STA A004
02BF A9 00    LDA #00
02C1 F0 09    BEQ 02CC
02C3 D0 0A    BNE 02CF
02C5 A9 86    LDA #86
02C7 8D 04 A0 STA A004
02CA A9 01    LDA #01
02CC 8D 05 A0 STA A005
02CF C6 EA    DEC EA
02D1 10 1A    BPL 02ED
02D3 A9 08    LDA #08
02D5 85 EA    STA EA
02D7 86 EB    STX EB
02D9 A6 E9    LDX E9
02DB E4 E8    CPX E8
02DD F0 0C    BEQ 02EB
02DF BD 00 03 LDA 0300,X
02E2 C9 C0    CMP #C0
02E4 F0 0C    BEQ 02F2
02E6 85 FE    STA FE
02E8 E8       INX
02E9 86 E9    STX E9
02EB A6 EB    LDX EB
02ED 68       PLA
02EE 2C 04 A0 BIT A004
02F1 40       RTI

02F2 A2 FF 9A A9 00 8D
02F8 0B A0 8D 0E A0 4C
02FE 00 00
```

Bild 2. Interrupt-Routine zur Ausgabe der Baudot-Zeichen im FIFO als fertiges FSK-(Frequency Shift Keying-)Signal am Port PB 7. Das Programmteil bei 02F2 dient zum Rückschalten in den Empfangsmodus und schaltet den Timer-Interrupt ab

Dazu wird die Interrupt-Routine in *Bild 2* verwendet. Sie bedient sich der beiden Timer im User-VIA 6522. Timer

1 wird im freilaufenden Modus betrieben und erzeugt die für die einzelnen Baudot-Bits erforderliche Tonfrequenz. Timer 2 bestimmt die Bitdauer bei der gewünschten Baudrate (hier 45 Bd).

Der Interrupt wird mit der doppelten erzeugten Tonfrequenz ausgelöst, bei 2125 Hz Nf-Ausgangssignal also 4250mal pro Sekunde. Um Phasensprünge zwischen den einzelnen Baudot-Bits zum vermeiden, wird Timer 1 nur dann mit dem momentan erforderlichen Frequenzwert geladen, wenn sich die Frequenz tatsächlich ändern muß.

Bild 3 zeigt die wichtigsten Adressen, die vom Programm benutzt werden. An der Adresse 0170 beginnt der Standardtext, der später mit der Taste F1 abgerufen werden kann. Er sollte nicht weiter reichen als etwa bis 01E0, um den Stackbereich nicht zu überschreiben, und muß mit 00 abgeschlossen werden. Der Nf-Ausgang ist PB 7 (Pin 15 des Application Connector); sein Ausgangssignal kann über einen Spannungsteiler (ca. 5000 : 1) der Sender-Mikrofonbuchse zugeführt werden. Besonders praktisch ist dabei ein Nf-gesteuerter Sender (VOX).

Das Programm ist für das Zusammenwirken mit dem bereits erwähnten Empfangsprogramm ausgelegt und funktioniert ohne dieses nicht. Die notwendigen geringfügigen Änderungen am Empfangsprogramm zeigt *Bild 4*. Soll vor jedem Buchstaben das BU-Umschaltzeichen ausgegeben werden, z. B. bei starken Störungen, so ist 0283 auf 00 zu ändern.

Inbetriebnahme

Sind Sende- und Empfangsprogramm geladen, so kann man den Mikrocomputer an der Adresse 0000 starten; er ist dann im Empfangsmodus. Der Drucker sollte dabei ausgeschaltet sein. Sobald die Taste S auf dem AIM-Tastenfeld gedrückt wird, schaltet sich das System in den Sendemodus und gibt zunächst 16 „Bu"-Zeichen aus. Während das geschieht, kann man bereits den zu sendenden Text eintippen. Dies kann fortlaufend ohne Drücken der Return-Taste geschehen, denn Wagenrücklauf- und Zeilenvorschub-Zeichen werden automatisch ausgestrahlt, wenn nach dem 50. Zeichen (vgl. 0247) einer Zeile ein Leerraum oder ein Bindestrich gefunden wird. Lediglich wenn man einen Absatz einfügen will, ist die Return-Taste zu betätigen.

```
BETRIEBSPARAMETER:
02A8 BAUDRATE (45 BD)
02BB MARK-FREQ.(2125 HZ)
02C6 SPACE-FREQ.(1275HZ)

ZERO-PAGE-ADRESSEN:
00E8 EINGABEZEIGER
00E9 AUSGABEZEIGER
00EA BITZAEHLER
00EB SAVE X
00FE ZEICHENBUFFER
```

Bild 3. Wichtige Adressen im Programm. Der FIFO-Speicher belegt den Bereich 0300...03FF, der Standardtext beginnt bei 0170

```
0000 A9 00 85 FD A9 BF 8D 80
0008 A4 2C 82 A4 70 03 4C 00
0010 02 A9 01 2C 0F A0 D0 EC
0088 5D
```

Bild 4. Notwendige Änderungen an dem bereits früher veröffentlichten Empfangsprogramm. Mit der Taste S kann man jetzt aus dem Empfangs- in den Sendemodus springen und mit F2 wieder zurück

Am Ende des zu sendenden Textes ist die Taste F2 zu drücken; sie sorgt dafür, daß der Mikrocomputer automatisch wieder in den Empfangsmodus schaltet, wenn das letzte Zeichen aus dem FIFO ausgegeben ist.

Der Aufruf des gespeicherten Standard-Textes erfolgt mit F1. Der Text wird aber nicht in das Display, sondern nur in den FIFO-Speicher transferiert, um dem Benutzer die Möglichkeit zu geben, sofort weiterzuschreiben. Dabei muß man darauf achten, daß die Taste F1 nicht zu oft nacheinander gedrückt wird, da sonst der 255 Zeichen umfassende FIFO-Speicher überläuft und nur noch Fragmente ausgibt.

Drückt man im Sendemodus die Escape-Taste, so erfolgt ein Sprung zum AIM-Monitorprogramm, ohne die FIFO-Ausgabe zu unterbrechen; d. h. der Interrupt läuft weiter, bis das letzte Zeichen ausgegeben ist, und springt dann zum Empfangsprogramm zurück.

Herwig Feichtinger

Ansteuerung des PET-Druckers

Mehrere Firmen vertreiben inzwischen Drucker für den Mikrocomputer PET 2001. Diese Peripheriegeräte werden an den IEC-Bus des PET angeschlossen. Leider liefert das PET-Handbuch nur spärliche Angaben über die Steuerung des Druckers per Software.

Man kann den Drucker auf zwei Arten ansprechen: Einmal durch Befehle innerhalb eines Programms, aber auch direkt, d.h. ohne Zeilennummer. In jedem Fall muß erst der PET seinen IEC-Bus „öffnen"; das wird erreicht durch OPEN 1,4. 1 ist dabei die laufende File-Nummer und 4 die Adresse des Druckers. Bei direkter Textausgabe oder für Programm-Listings folgt dann der Befehl CMD1; die 1 bezieht sich wieder auf die File-Nummer.

Jetzt gibt man ein: PRINT # 1, „Text" oder PRINT # 1, LIST usw. Der File muß anschließend wieder mit CLOSE 1 geschlossen werden. Beim Ausdrucken von Text und Variablen innerhalb eines Programmes kann CMD entfallen.

Hier gleich ein paar Beispiele. Ein Programm-Listing kann man entweder anfertigen, indem man vor das Programm die folgende Zeile schreibt:

1 OPEN 1,4 : CMD1 : LIST

Oder aber man gibt die notwendigen Befehle direkt ein:

OPEN 1,4 RETURN
CMD1 RETURN
(die Adresse ist nicht immer 4!)

Der PET meldet sich dann über den Drucker mit READY; der Befehl LIST führt dann zum Ausdruck des Programms. Dann schreibt man CLOSE 1, um den IEC-Bus wieder zu schließen.

Innerhalb eines Programmes kann man auch über den Drucker Text ausgeben, z. B. so:

```
10 OPEN 1,4: CMD1
20 PRINT#1, „ICH BIN EIN PET"
30 CLOSE 1
```

Genauso funktioniert im Prinzip der Ausdruck von Zahlenwerten oder Variablen.

Den Wagenrücklauf machen die meisten Drucker mit dem ASCII-Zeichen (hex) 0D, den Zeilenvorschub mit hex 0A. Beide Befehle sind im PET nicht vorgesehen, sie lassen sich aber erzeugen, indem man Strings definiert:

```
A$ = CHR$ (10)
PRINT#1, A$, A$
```

Diese Befehlsfolge ergibt einen dreimaligen Papiervorschub, da der PET bereits bei PRINT einen Vorschub ausführt.

Engelbert Misera

Rudolf Hofer

Datenbank-Programm für den TRS-80

Eine der Standardaufgaben für Computer besteht darin, Daten abzuspeichern und sie dem Benutzer bei Bedarf wieder zugänglich zu machen. Das ideale Speichermedium für private Zwecke ist in diesem Fall die Mini-Diskette: Sie nimmt einerseits genügend Information auf (rund 40 Schreibmaschinenseiten) und ist andererseits hinreichend schnell zu durchsuchen. Mit dem vorgestellten Programm, das in BASIC geschrieben wurde, sind TRS-80-Besitzer mit Floppy-Disk-Laufwerk in der Lage, sich eine kleine Datenbank anzulegen.

Betriebsarten

Das Programm kennt fünf Betriebsarten, von denen im Normalfall zwei benutzt werden. In der einen – „Fortlaufend schreiben" – kann der Benutzer Records bis zu einer Länge von 248 Zeichen (auch alle Sonderzeichen sind erlaubt) in die Datei einschreiben. Als Record bezeichnet man ein abgeschlossenes „Datenpaket" – etwa eine Adresse –, das vom Programm als Ganzes behandelt wird. Wo der Record abgelegt wird, braucht den Benutzer nicht zu kümmern – mit einer Ausnahme: wenn die Datei zum ersten Mal mit Daten beschrieben wird.

Die zweite Haupt-Betriebsart nennt sich „Suchen". Sie erfragt vom Benutzer zwei Begriffe und durchsucht dann alle abgespeicherten Records, ob einer der beiden vorkommt oder ob beide vorkommen. Welches Kriterium gewählt wird, bestimmt wiederum der Benutzer. Die gefundenen Records werden ausgedruckt. Als Begriff ist jede beliebige Zeichenfolge bis zu einer Länge von 255 Zeichen erlaubt.

Mit zwei weiteren Betriebsarten können logische Records, deren Bedeutung noch näher erklärt wird, gezielt gelesen und geschrieben werden. In der Praxis verwendet man sie zum Ändern von Records.

In der fünften Betriebsart schließlich teilt man dem Computer mit, wo er bei „fortlaufendem Einschreiben" den nächsten Record ablegen soll. Wie erwähnt, ist das nur ein einziges Mal nötig. Der Computer merkt sich diesen Wert, auch wenn er ausgeschaltet wird, und bringt ihn laufend auf den neuesten Stand.

Bedienung

Wird das Programm zum ersten Mal in Betrieb genommen, startet man es mit RUN 790. Auf dem Bildschirm erscheint dann die Frage LETZTER RECORD? Zweckmäßigerweise gibt man die Zahl 1 ein (nicht vergessen: nach jeder Eingabe Enter-Taste drücken). Die Daten werden dann ab dem logischen Record 2 eingeschrieben. Im logischen Record 1 könnte man den Namen der Datei ablegen, falls es nicht die einzige ist.

Von jetzt ab wird das Programm nur noch mit RUN gestartet. Der Computer löscht daraufhin den Bildschirm und druckt folgenden Text aus:
1 = SCHREIBEN 2 = LESEN 3 = FORTLAUFEND SCHREIBEN 4 = SUCHEN?
Er erwartet dann eine der Zahlen 1...4, mit denen der Benutzer die entsprechende Betriebsart wählt. Bevor die verschiedenen Möglichkeiten näher besprochen werden, soll der Begriff „logischer Record" kurz erklärt werden:

Normalerweise ist die kleinste Datenmenge, die als Record von und zur Diskette transportiert werden kann, 256 Byte. Um nicht zu viel Platz zu verschenken, werden per Programm kleinere „Portionen" zu je 62 Byte benutzt. Sie werden der Reihe nach durchnumeriert und als logische Records bezeichnet.

Fortlaufend schreiben

Der Computer druckt die Nummer des logischen Records aus, der als nächster eingeschrieben wird, und fordert dann den Benutzer auf: TEXT EINGEBEN. Gibt man lediglich eine Null ein, kehrt das Programm zum Anfang zurück, und man kann eine neue Betriebsart wählen. Ein abzuspeichernder Text wird mit der Enter-Taste abgeschlossen, worauf der nächste Record angefordert wird. Vorher kann innerhalb einer Zeile korrigiert werden. Für Wagenrücklauf/Zeilenvorschub wird die Taste „↓"verwendet.

Das Programm ist so aufgebaut, daß vorzugsweise kurze Records mit weniger als 62 Zeichen abgespeichert werden. Längere müssen beim Suchlauf erst miteinander verbunden werden, und das kostet sehr viel Zeit. Trotzdem darf der Benutzer Texte bis zu 248 Zeichen eingeben. Sind sie jedoch im allgemeinen länger als 62 Byte, dann empfiehlt sich eine kleine Programmkorrektur, die einem logischen Record mehr Platz zur Verfügung stellt.

Suchen

Der Computer druckt die Frage BEGRIFFE? aus und erwartet zunächst den ersten Suchbegriff. Mit einer Null kehrt man wieder zum Programmanfang zurück und kann eine neue Betriebsart wählen. Wird ein Suchbegriff eingegeben, so schließt man ihn mit Enter ab und tippt den zweiten ein. Mit einer Eins sagt man dem Computer, es sollen beide Begriffe im Record vorkommen (UND-Verknüpfung); eine Zwei bedeutet, es genügt, wenn ein Begriff vorkommt (ODER-Verknüpfung).

Soll nur nach einem Begriff gesucht werden, kommt der Computer am schnellsten zum Ziel, wenn man dieses Wort als ersten Suchbegriff verwendet, beim zweiten lediglich Enter drückt und sich dann für UND entscheidet. Sind die Daten gefunden, fragt der Computer nach weiteren Begriffen.

Gezielt lesen

Auf die Frage LOG. REC.-NR.? tippt der Benutzer eine Zahl ein, und der entsprechende logische Record wird gelesen. Diese Betriebsart wird man vor allem dann verwenden, wenn man anschließend einen Record ändern will. Im Lese-Modus ist man so lange, bis eine Null eingegeben wird.

Gezielt schreiben

Auch hier gibt man zuerst die Nummer des logischen Records ein. Ist sie Null, kehrt man zum Programmanfang zurück und kann eine neue Betriebsart wählen. Andernfalls fordert der Computer auf: TEXT EINGEBEN. Achtung: Jetzt kann tatsächlich nur ein logischer Record eingegeben werden, d. h. maximal 62 Zeichen. Nachdem man die Eingabe mit der Enter-Taste abgeschlossen hat, erscheint die Frage TEXT BEENDET = ENTER? auf dem Bildschirm.

Bei erneutem Enter kann ein weiterer logischer Record geschrieben werden. Hat der Platz von 62 Zeichen aber nicht ausgereicht, um die gewünschte Information einzugeben, dann drückt man an dieser Stelle nicht die Enter-Taste, sondern tippt die niederwertige Stelle der logischen Record-Nummer ein. Diesselbe Zahl wird bei den unmittelbar folgenden Records eingegeben, sofern sie inhaltlich zum vorhergehenden Text gehören. Maximal dürfen vier logische Records auf diese Weise verknüpft werden.

Ein Beispiel soll diesen Vorgang verdeutlichen: Nehmen wir an, wir hätten festgestellt, die Adresse in den logischen Records 32 und 33 hat sich geändert. Wir wählen die Betriebsart „Gezielt schreiben" und geben die logische Record-Nummer 32 ein. Jetzt verbessern wir die ersten 62 Zeichen der Adresse und drücken die Enter-Taste. Auf dem Bildschirm erscheint TEXT BEENDET = ENTER? Da sich die Änderung auch noch auf den logischen Record 33 bezieht, geben wir jetzt 2 (= niederwertige Stelle der log. Record-Nummer) ein und drücken dann erst Enter. Wieder fragt der Computer nach der log. Record-Nummer. Wir geben 33 ein und setzen dann unsere Korrektur fort. Nach Enter erscheint wieder TEXT BEENDET = ENTER? Jetzt wird noch

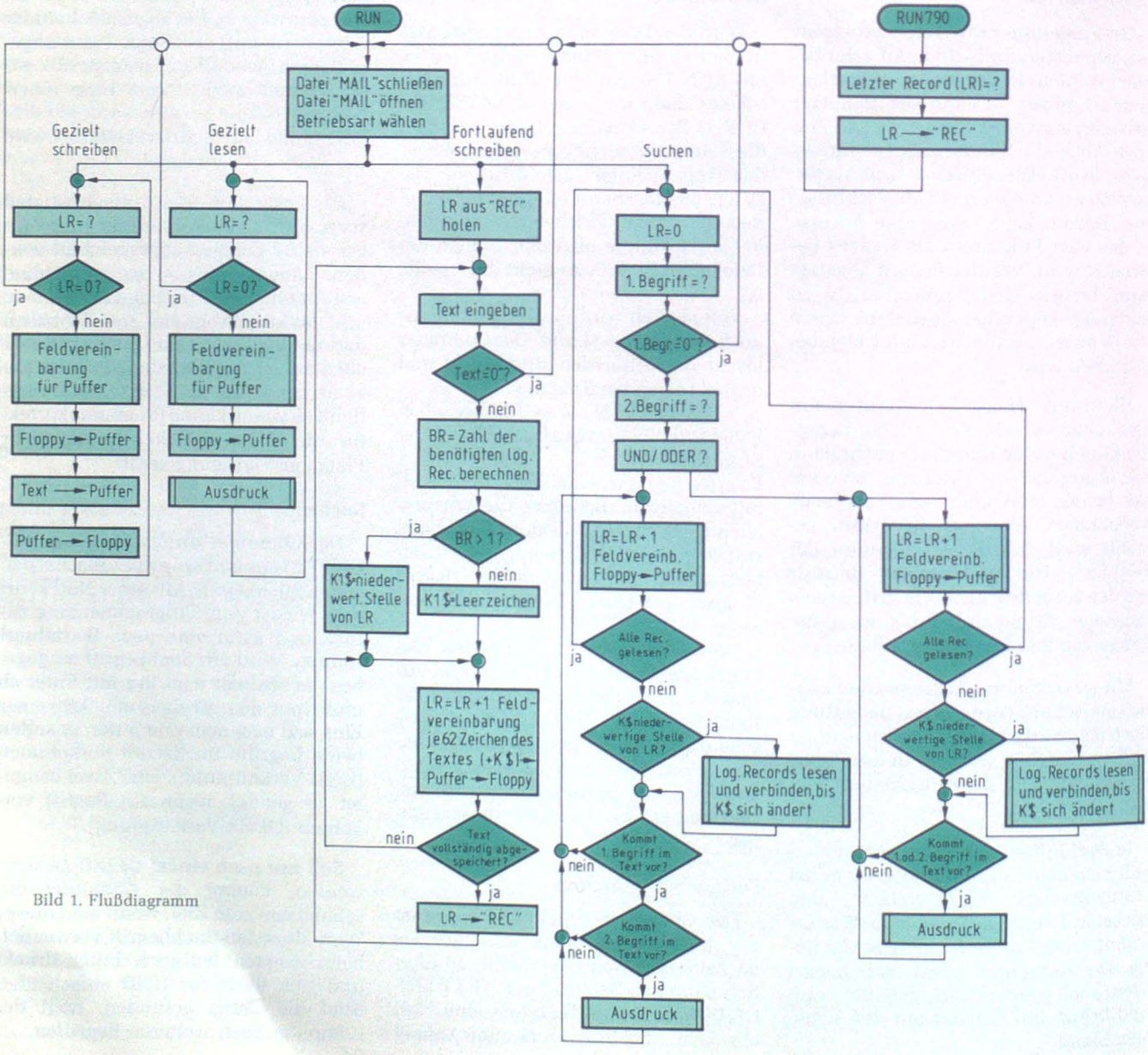


Bild 1. Flußdiagramm

```
99 REM KOMMANDOSCHLEIFE
100 CLEAR1000:CLS:CLOSE
110 PRINT:INPUT"1=
SCHREIBEN 2=LESEN
3=FORTLAUFEND SCHREIBEN
4=SUCHEN";N
120 OPEN"R",1,"MAIL"
130 CLS:ON N GOTO200
,300,400,560
199 REM GEZIELT
SCHREIBEN
200 PRINT:INPUT"LOG.
RECORD-NR.";LR
210 IF LR=0 THEN100
220 GOSUB2000
230 GET1,PR
235 PRINT"TEXT EINGEBEN"
240 LINE INPUT "";A$:LSET
N$=A$
250 INPUT"TEXT
BEENDET=ENTER";A$:LSET
K$=A$
280 PUT 1,PR:GOTO200
299 REM GEZIELT LESEN
300 PRINT:INPUT"LOG.
RECORD-NR.";LR
310 IF LR=0 THEN 100
320 GOSUB2000
330 GET 1,PR
340 GOSUB3100
350 GOTO300
399 REM FORTLAUFEND
SCHREIBEN
400 OPEN"I",2,"REC"
410 INPUT#2,LR
420 CLOSE 2
475 PRINT"LOG.
REC.-NR.="; LR+1:PRINT:
PRINT"TEXT
EINGEBEN":PRINT
480 LINE INPUT"";
A$:Z=LEN(A$)
481 IF A$="0" THEN 100
482 BR=INT(Z/62+0.99):
N=0
483 IF BR>1 THEN
K1$=RIGHT$(STR$(LR+1),1)
ELSE K1$=" "
484 GOSUB3000
485
A1$=MID$(A$,N*62+1,62):
LSET N$=A1$:LSET K$=K1$
488 N=N+1
490 PUT 1,PR
492 IF N<BR THEN 484
493 OPEN"O",2,"REC"
494 PRINT#2,LR
496 CLOSE 2
499 GOSUB 2000
530 GOTO 475
559 REM SUCHEN
560 INPUT"BEGRIFFE";C$
565 LR=0
570 IF C$="0" THEN100
575 INPUT F$
580 INPUT"UND=1
ODER=2";N
590 ON N GOTO 600,700
600 GOSUB 3000
615 IF LOF(1)*4<LR
THEN560
617 IFK$=RIGHT$(STR$(LR)
,1) GOSUB 3300
620 IF INSTR(N$,C$)=0
THEN 600
630 IF INSTR(N$,F$)=0
THEN 600
640 GOSUB 3100
650 GOTO600
700 GOSUB 3000
715 IF LOF(1)*4<LR
THEN560
717 IFK$=RIGHT$(STR$(LR)
,1) GOSUB 3300
720 IF INSTR(N$,C$)=0 AND
INSTR(N$,F$)=0THEN 700
730 GOSUB 3100
740 GOTO 700
769 REM LETZTEN RECORD
FESTLEGEN
790 INPUT"LETZTER
RECORD";LR
800 OPEN"O",2,"REC"
810 PRINT#2,LR
820 CLOSE2
830 GOTO100
1999 REM UNTERPROGRAMM
FUER FELDVEREINBARUNG
2000 SR=LR-4*INT((LR-1)
/4)-1
2020 FIELD 1,SR*63 AS
D$,62 AS N$,1 AS K$
2030 PR=INT((LR-1)/4)+1
2040 RETURN
2999 REM LESEN
3000 LR=LR+1
3010 GOSUB2000
3020 GET 1,PR
3030 RETURN
3099 REM AUSDRUCKEN
3100 CLS:PRINT:PRINT
3105 PRINT"LOG.
REC.-NR=";LR
3107 PRINT:PRINT
3110 PRINT N$
3130 PRINT:RETURN
3299 REM VERBINDEN
3300 K1$=K$:M$=CHR$(0)
3310 M$=M$+N$
3320 GOSUB 3000
3330 IF K$=K1$ THEN 3310
3340 N$=M$
3350 LR=LR-1:RETURN
```

Bild 2. Das Listing wurde mit Hilfe eines Formatier-Programmes ausgegeben, das die Zeilenlänge beschränkt. Jede Zeile, die nicht mit einer Zahl beginnt, ist deshalb an die vorhergehende anzuschließen. Ob ein Leerzeichen eingefügt wird, ist im allgemeinen ohne Bedeutung

einmal 2 eingetippt und mit Enter abgeschlossen. Nun weiß der Computer, daß die logischen Records 32 und 33 zusammengehören. Um in eine neue Betriebsart zu gelangen, beantworten wir die neuerliche Frage nach der log. Record-Nummer mit Null.

Programmdetails

Die vier Hauptprogrammteile bestehen aus Schleifen, die durch den ON-Befehl in Zeile 130 wahlweise angesprungen werden und die man mit einer Null als Antwort auf die jeweils erste Frage wieder verlassen kann (*Bilder 1* und *2*). Wesentlicher Bestandteil ist das Unterprogramm für die Feldvereinbarung ab Zeile 2000, das für die richtige Plazierung der logischen Records innerhalb der physikalischen Records sorgt. Es berechnet bei bekanntem LR die Werte PR und SR, wie es in [1] beschrieben ist. Damit braucht man im übrigen Programm nur noch mit logischen Record-Nummern (LR) zu arbeiten.

Außer der eigentlichen (wahlfreien) Datei „MAIL" wird eine zweite (sequentielle) verwendet, die mit „REC" bezeichnet wird. Sie enthält die Nummer des letzten beschriebenen logischen Records und wird bei „fortlaufendem Schreiben" automatisch inkrementiert.

Um sich das Prinzip klarzumachen, betrachtet man sich am besten die Programmteile „Gezielt schreiben" (200...280) und „Gezielt lesen" (300...350), die geradlinig und ohne Hinterhältigkeiten ablaufen. Erweiterungen davon sind die Betriebsarten „Fortlaufend schreiben" und „Suchen".

Betrachten wir uns erstere für den Fall, daß ein Text mit weniger als 63 Zeichen abzuspeichern ist. Die Zahl der benötigten logischen Records BR ist dann 1. K1$ ist eine Hilfs-Variable, deren Wert später K$ zugewiesen wird. K$ aber ist ein Zeichen, das mit jedem logischen Record abgespeichert wird und Auskunft darüber gibt, ob der nächste logische Record mit dem vorhergehenden zu verbinden ist. Bei Texten mit weniger als 63 Zeichen erhält K$ das Leerzeichen zugewiesen. Der Text ist also im betrachteten Fall nach dem ersten Durchlauf vollständig abgespeichert, und weitere Daten können eingegeben werden.

Hat der Text mehr als 62 Zeichen, wird er in mehreren logischen Records abgespeichert, und K$ erhält jeweils den Wert der niederwertigen Stelle der ersten logischen Record-Nummer (*Bild 3*).

Beim Suchen werden zuerst die beiden Begriffe eingegeben, dann verzweigt das Programm, abhängig davon, ob man sich für UND oder ODER entschieden hat. Diese beiden Teile unterscheiden sich allein dadurch, daß bei ODER jedesmal beide Begriffe in einem Record gesucht werden müssen, während bei UND zuerst nach einem gesucht wird und erst bei Erfolg nach dem zweiten. Das wirkt sich auf die Geschwindigkeit aus, weshalb man immer UND wählen sollte, wenn man nur ein Suchwort hat (siehe Bedienung).

Bild 3. Zusammengehörige Texte werden durch den Wert von K$ ausgedrückt: Falls ein logischer Record (LR) für sich steht, wird ein Leerzeichen abgespeichert, andernfalls erhält K$ die niederwertige Stelle des ersten LR zugewiesen. Dies ist willkürlich festgelegt; wichtig ist allein, daß man ein Kriterium erhält, mit dem man beim Suchen die Records wieder richtig verbinden kann

LR	Text (N$)	K$
11	Text 1	
12	Text 2	
13	Text 3	3
14	Text 3	3
15	Text 3	3
:	:	
23	Text 7	
24	Text 8	4
25	Text 8	4
26	Text 9	
27	Text 10	7
28	Text 10	7

Da man beim Einschreiben die Möglichkeit hat, den Text einzugeben, ohne auf die Record-Grenzen zu achten, kann es vorkommen, daß man ein Wort in der Mitte trennt. Damit auch dieses Wort als Ganzes wieder gefunden wird, muß man zusammengehörende Records verbinden, bevor nach einer Zeichenfolge gesucht wird. Dies besorgt das Unterprogramm ab Zeile 3300, das abhängig vom Wert der Variablen K$ angesprungen wird (*Bild 3*). Da diese zusätzlichen Befehle sehr zeitaufwendig sind, sollte man möglichst darauf verzichten, sehr oft längere Records zu verwenden.

Wer im allgemeinen längere Texte abspeichern will, kann folgende Änderungen vornehmen, die das Programm für einen durchschnittlichen Bedarf von knapp 126 Zeichen optimieren:

Zeile 482: 62 durch 126 ersetzen, 0.99 durch 0.999 ersetzen
Zeile 486: 62 zweimal ersetzen durch 126
Zeilen 615, 715 und 2000: Jede 4 ersetzen durch 2
Zeile 2020: 63 ersetzen durch 127, 62 ersetzen durch 126
Zeile 2030: 4 ersetzen durch 2

Der Zeitbedarf für das Suchen kann nur grob angegeben werden. Er beträgt etwa 200 ms/log. Record, wenn die empfohlene Länge nicht überschritten wird (bei Überschreiten etwa das Vierfache). Das Programm belegt insgesamt 1600 Byte RAM.

Literatur

[1] *Hofer, R.*: Dateiverarbeitung mit dem TRS-80. FUNKSCHAU 1979, H. 15, S. 887.

```
0000 A9 00     LDA #00
0002 85 FA     STA FA
0004 A9 02     LDA #02
0006 85 FB     STA FB
0008 A2 05     LDX #05
000A BD D7 1F  LDA 1FD7,X
000D 20 00 01  JSR 0100
0010 CA        DEX
0011 D0 F7     BNE 000A
0013 A4 EE     LDY EE
0015 B1 FA     LDA (FA),Y
0017 C9 21     CMP #21
0019 30 05     BMI 0020
001B 88        DEY
001C D0 F7     BNE 0015
001E A4 EE     LDY EE
0020 84 F3     STY F3
0022 20 63 1F  JSR 1F63
0025 A0 00     LDY #00
0027 B1 FA     LDA (FA),Y
0029 D0 03     BNE 002E
002B 4C 4F 1C  JMP 1C4F
002E C9 0D     CMP #0D
0030 F0 D6     BEQ 0008
0032 20 00 01  JSR 0100
0035 C6 F3     DEC F3
0037 D0 E9     BNE 0022
0039 F0 F5     BEQ 0030
003B 00        BRK
```

Bild 1. Hauptprogramm zum Ausdruck gespeicherter Texte

Textausdruck mit Baudot-Fernschreiber

In Heft 24/1978 wurde für den Mikrocomputer KIM-1 (µP: 6502) unter dem Titel „KIM auf Datensuche" ein Karteiprogramm vorgestellt, das die Speicherung von Adressen, Telefonnummern oder anderen Texten und Daten gestattet. Ein preiswerter gebrauchter Fernschreiber ermöglicht mit dem nachfolgenden Programm den Ausdruck einer solchen Kartei.

Weil jeder einzelne Informationsblock länger als eine Fernschreiber-Zeile sein kann, nämlich bis zu 255 Zeichen, muß das Druckprogramm auch dafür sorgen, daß rechtzeitig am Ende einer Zeile eine neue Zeile begonnen wird. Um dies zu bewerkstelligen, ohne ein Wort zu „zerreißen", wird derjenige Leerraum ermittelt, der sich – je nach gewünschter Zeilenlänge – für die Trennung eignet.

Bild 1 enthält das Hauptprogramm, das ein Unterprogramm zur Ausgabe eines Baudot-Zeichens (*Bild 2*) aufruft. Dieses Unterprogramm verwendet wiederum eine ASCII-Baudot-Umwandlungstabelle, die in *Bild 3* aufgelistet ist. Die beiden letztgenannten Programmteile stimmen weitgehend mit dem Programm aus Heft 1/1979 überein und wurden daher nur in hexadezimaler Form wiedergegeben.

Die gewünschte Zeichenzahl pro Zeile muß in die Zelle 00EE eingegeben werden, und zwar als hexadezimaler Wert; für 62 Zeichen pro Zeile muß 00EE also hex 3E enthalten.

Für diejenigen Leser, die keinen KIM-1 besitzen, sei noch die Bedeutung der Monitor-Adressen erwähnt. An der Adresse 1FD7 befindet sich die Zeichenfolge 00 00 00 00 0A 0D, die zur Ausgabe von Wagenrücklauf und Zeilenvorschub dient. Die Null-Zeichen sollen dem Wagen des Fernschreibers genügend Zeit zum Rücklauf geben. Die Adresse 1C4F ist die Startadresse des Monitorprogramms, zu der bei Beendigung des Druckvorganges gesprungen wird, und bei 1F63 steht ein ROM-Unterprogramm, das den Textzeiger 00FA, 00FB (16-bit-Zeiger) um 1 erhöht.

Herwig Feichtinger

Berichtigung

Roulett auf dem TI-59

FUNKSCHAU 1979, Heft 12, Seite 717

Aufgrund eines Magnetkarten-Lesefehlers wurden mehrere Programmschritte falsch reproduziert. Richtig muß es heißen:

212	15	E	231	19	19
221	01	1	232	43	RCL
224	77	GE	233	07	07
225	42	STO	234	55	DIV
226	43	RCL	235	02	2
227	17	17	236	95	=
228	76	LBL	237	22	INV
229	55	DIV	238	59	INT
230	44	SUM	239	29	CP

Wie solch ein Fehler passiert, ist im gleichen Heft unter dem Titel „TI-59-Magnetkarten nicht austauschbar" nachzulesen.

Bild 2. ▶ Unterprogramm zur Ausgabe eines ASCII-Zeichens auf einem Baudot-Fernschreiber. Er ist über eine Treiberstufe (z. B. Darlington-Transistor) an den KIM-Port PA 7 anzuschließen

```
0100 86 F5 A2 FF 8E 01 17 C9
0108 0D D0 04 A9 08 D0 0E C9
0110 0A D0 04 A9 02 D0 06 29
0118 3F AA BD 70 01 85 F6 29
0120 20 C5 F7 F0 0E 85 F7 A8
0128 F0 04 A9 1B D0 02 A9 1F
0130 20 3D 01 A5 F6 20 3D 01
0138 A0 FF A6 F5 60 A2 00 8E
0140 00 17 20 64 01 A0 05 4A
0148 90 04 A2 FF D0 02 A2 00
0150 8E 00 17 20 64 01 88 D0
0158 EE A2 FF 8E 00 17 A2 1E
0160 20 66 01 60 A2 14 8E 07
0168 17 2C 07 17 10 FB 60 00
```

```
0170 00 03 19 0E 09 01 0D 1A 14 06 0B 0F 12 1C 0C 18
0180 16 17 0A 05 10 07 1E 13 1D 15 11 2D 00 3A 00 00
0190 04 34 00 2C 00 24 31 25 2F 32 39 31 2C 23 3C 3D
01A0 36 37 33 21 2A 30 35 27 26 38 2E 2B 24 3E 24 39
```

Bild 3. Codeumwandlungstabelle für das Baudot-Unterprogramm

Hans Werner

Mikrocomputer

Eine kleine Einführung

23. Teil

Soll das Übersetzungsprogramm selbst die absoluten Adressen suchen, ist die Adressenreservierung m überflüssig.

```
     ORG     OD
     .
     .
     BR EX   M
     ORG     300D
S    VARI
Z    VARI
     END
```

— Neue ORG-Anweisung, damit S und Z im RAM reserviert werden (Adresse größer 256).

In dieser Form ist das Programm im ROM bzw. RAM gespeichert. Alle Mnemonics, Adressen und Konstanten sind durch binäre Muster ersetzt.
Z. B. ist der Operationsteil von SP AC d S als 00000011 gespeichert. Für S steht die absolute Adresse 00000001 00101100 (300).

Maschinenprogramm

	SPEICHER ROM ADRESSE	INHALT	ASSEMBLER
0	00000000 00000000	01000001	LA AC k
1	0 00000001	00000000	OD
2	0 00000010	00000011	SP AC d
3	0 00000011	00000001	S
4	0 00000100	00101100	
5	0 00000101	01000010	LA IR k
6	0 00000110	01100100	100D
7	0 00000111	01000001	LA AC k
8	0 00001000	11001000	201D
9	0 00001001	00000011	SP AC d
10	0 00001010	00000001	Z
11	0 00001011	00101101	
12	0 00001100	00000001	M LA AC d
13	0 00001101	00000001	S
14	0 00001110	00101100	
	. . .	. . .	
27	0 00011011	00010010	BR EX d
28	0 00011100	00000000	M
29	0 00011101	00001100	
30	00000000 00011110	00011100	END
	SPEICHER RAM		
300	00000001 00101100	wechselt	S
301	00000001 00101101	wechselt	Z

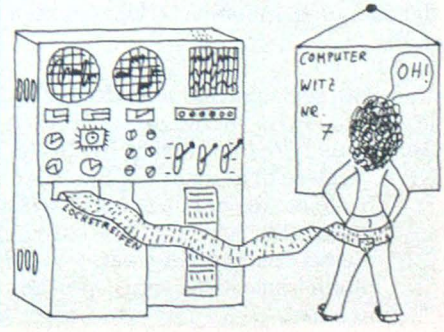
Sprungziel und Marke M erhalten nach dem Übersetzen die laufende Adresse 00000000 00001100 (12) usw.
Bemerkung: Die Bitmuster für den Operationsteil der Befehle sind willkürlich gewählt und sollen natürlich nicht erlernt werden. Das wäre Unsinn.

Hexadezimale Schreibweise

Bei der Erprobung eines Programms wird meist nicht der eigentliche Mikrocomputer benutzt, sondern ein sogenanntes Entwicklungs-System.
Es enthält zwar auch einen Mikrocomputer des gewählten Typs, aber zusätzlich Dienstprogramme zur Fehlersuche und Assemblierung. Weiterhin können Sichtgeräte und Drucker angeschlossen werden.

ADRESSE	INHALT	ASSEMBLER
00 00	41	LA AC k
00 01	00	OD
00 02	03	SP AC d
00 03	01	S
00 04	2C	
. .	. .	
01 2D	wechselt	Z

Will der Programmierer sich das Maschinenprogramm ansehen, läßt er es in hexadezimaler Schreibweise ausdrucken.
Bei einiger Übung ist das übersichtlicher als die binäre Form.



BEFEHLE			ADRESSIERUNG		
	ERKLÄRUNG	KÜRZEL	DIREKT	UNMITTELBAR	INDIZIERT
1	Lade Akku	LA AC n	(n) in AC	n in AC	((IR)+n) in AC
2	Lade IR	LA IR n	(n) in IR	n in IR	" in IR
3	Speichere aus Akku	SP AC n	(AC) nach n	—	(AC) nach (IR)+n
4	Speichere aus IR	SP IR n	(IR) nach n	—	(IR) nach "
5	Addieren	AD n	(AC)+(n) in AC	(AC)+n in AC	(AC)+((IR)+n) in AC
6	Subtrahieren	SU n	" - "	" - "	" - "
7	UND	UN n	" UND "	" UND "	" UND "
8	ODER	OD n	" ODER "	" ODER "	" ODER "
9	Negieren (1-er Kompl.)	NE n	(n) nach n		((IR)+n) nach (IR)+n
10	Inkrement von n	IN n	(n)+1 nach n	—	((IR)+n)+1 nach (IR)+n
11	Dekrement von n	DE n	(n)-1 nach n	—	" -1 "
12	Inkrement von IR	IN IR	(IR)+1 in IR	—	—
13	Dekrement von IR	DE IR	" - "	—	—

	ERKLÄRUNG		KÜRZEL	DIREKT	UNMITTELBAR
14	rechts schieben		SH RE n	—	um n
15	links schieben		SH LI n	—	
16	unbedingter Sprung		JMP n	mit n weiter	—
17	Springe wenn : (Branch)	Ergebnis = 0	BR EO n		—
18		Ergebnis ≠ 0	BR EX n		—
19		Carry = 0	BR C0 n		—
20		Carry = 1	BR C1 n		—
21		Ergebnis pos	BR EP n		—
22		Ergebnis neg	BR EN n		—
23	Unterprogramm-Aufruf		RUF n	—	—
24	Zurück in das Hauptpr.		POP	—	—
25	Setze Interrupt-FF		SET I n	—	n in FF
26	Setze Carry -FF		SET C n	—	n in FF
27	Übertrag in den Akku		CA AC	—	—
28	Ruhezustand, Ende		END	—	—

Adressierung
direkt : d anhängen, zB. LA AC d
unmittelbar: k anhängen, zB. LA AC k
indiziert : i anhängen, zB. LA AC i

Befehle 2,4,9,10,11: Fehlt bei der direkten Adressierung n, wird der Akku angesprochen.
Befehle 16 ... 22: Die indizierte Adressierung ist nicht in der Tabelle enthalten.

Befehle 25,26: Nur 0 und 1 erlaubt
Klammern : symbolisieren den Inhalt eines Registers oder einer Speicherzelle.

Befehlsliste

So sieht die vollständige Befehlsliste unseres Modells aus. Es könnte auch die Liste eines richtigen Mikrocomputers sein.

1...4
sind Transportbefehle, die den Datentransport zwischen Akkumulator bzw. Indexregister und Speicher ermöglichen. Unbekannt ist uns noch Befehl 4, der aber nicht gesondert erklärt werden muß.

5...13
sind arithmetische und logische Befehle. Zu ihnen ist wenig zu sagen. Eine Ausnahme bildet der Befehl NEGIERE.

NE n: Negiere den Inhalt der Speicherzelle n. Aus z. B. 11100011 wird 00011100.
Selbstverständlich darf keine Zelle im ROM angesprochen werden.

NE: Negiere den Inhalt des Akkumulators (siehe Bemerkungen unter der Liste).

14...15
Rechenwerkbefehle, die sich nur auf den Akkumulator auswirken.

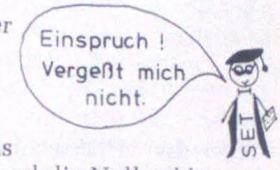
16...28
Organisationsbefehle.
Mit ihnen können Entscheidungen getroffen (Sprungbefehle), Unterprogramme organisiert und das Carry-Flip-Flop beeinflußt werden. Die Befehle 23...26 stellen wir vorläufig zurück und wenden uns den Sprungbefehlen zu.

Unbedingter Sprung
JP n: Springe auf jeden Fall nach n.

Bedingte Sprünge
Die Ausführung des Sprungbefehls ist abhängig vom Inhalt des Statusregisters.

a) wenn das Ergebnis einer Rechenoperation 0 bzw. ungleich 0 ist (BR EO, BR EX).
b) wenn ein Übertrag (Carry) auftritt bzw. nicht auftritt (BR C1, BR C0).
c) wenn das Ergebnis positiv bzw. negativ ist (BR EP, BR EN).
Akku-Bit 7 = 0: positiver Wert
1: negativer Wert
Das gilt natürlich auch dann, wenn mit vorzeichenlosen Zahlen gerechnet wird.

Statusregister



Die Befehle 5...13 beeinflussen das Carry-Bit, das Vorzeichen und die Nullmeldung. Die Befehle 14 und 15 lediglich Vorzeichen und Nullmeldung, während das Carry-Bit stets zu Null wird. Mit SET C kann das Carry-Bit auf 0 oder 1 gesetzt werden. Alle anderen Befehle haben keinen Einfluß auf das Statusregister.

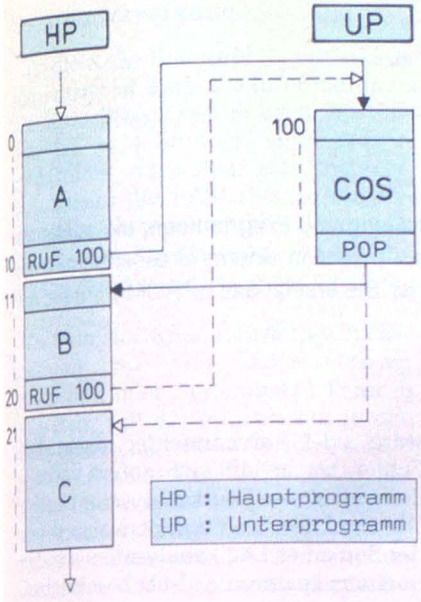
Indexregister
Im Gegensatz zum Datenregister kann das Indexregister gezielt vom Programmierer benutzt werden. Es ändert seinen Inhalt nur durch die Befehle LA IR, DE IR und IN IR.
Das ist nicht bei allen Mikrocomputern der Fall, z. B. wird es beim Typ F8 automatisch mit jedem Befehl inkrementiert. Das Programm „Indiziale Befehle" vereinfacht sich dann um einen Befehl.

Vereinbarung
Die Kennzeichen d, k bzw. i können entfallen, wenn ein Befehl nur eine Adressiermöglichkeit zuläßt.
Beispiel: DE IR

Unterprogramme

Häufig tritt diesselbe Befehlsgruppe an verschiedenen Stellen des Programms auf. Solche Wiederholungen werden durch Unterprogramme vermieden. Sie können beliebig oft aufgerufen und abgearbeitet werden. Das Programm verkürzt sich, es wird Speicherplatz und Schreibarbeit gespart. Die Laufzeit des Programms bleibt in etwa erhalten.

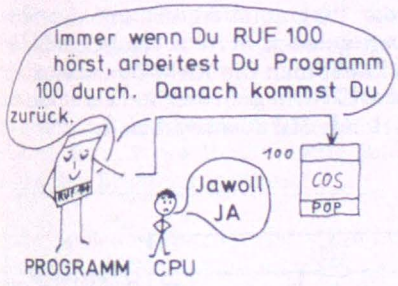
Beispiel:
Ein Programm besteht aus den Teilen A, B und C. Nach den Abschnitten A und B ist der Kosinus des jeweiligen Ausgangswertes zu ermitteln. Um das Kosinus-Programm nur einmal zu schreiben, wird es als Unterprogramm organisiert. Es steht ab Adresse bzw.



Marke 100 im Speicher. Unterprogramme werden bei unserem Modell mit der Anweisung RUF aufgerufen.

Befehl 10:
Die Anweisung RUF spricht das Unterprogramm an. Es wird bis zum letzten Befehl POP durchgearbeitet, der das Auffinden der Rücksprungadresse 11 ermöglicht.

Befehl 20:
Erneuter Aufruf des Unterprogramms und Rückkehr zur Anschlußadresse 21.



Problematisch ist der Rücksprung in das Hauptprogramm. Im Gegensatz zum normalen Sprungbefehl ändert sich die Zieladresse nach jedem Aufruf (11, 21).

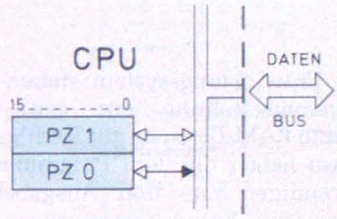
Mit normalen Sprungbefehlen sind deshalb Unterprogramme nur mit viel Programmieraufwand möglich. Das umständliche Umrechnen der Rücksprungadresse mag der Leser selbst versuchen.

Unterprogramme (Subroutines) können innerhalb des übergeordneten Hauptprogramms mehrfach benutzt werden. Sie müssen mit der Anweisung POP enden. Es dürfen beliebig viele Unterprogramme in das Hauptprogramm eingefügt werden.

Das gilt nicht für verschachtelte Unterprogramme, d. h. wenn innerhalb eines Unterprogramms ein weiteres aufgerufen wird. Solche Anordnungen werden später erläutert.

Ablauf

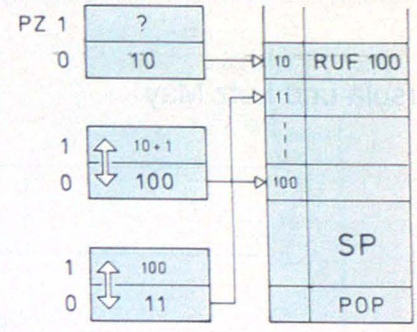
In der CPU gibt es noch den zweiten Programmzähler PZ 1. Bisher wurde nur PZ 0 benutzt, der in jedem Befehlszyklus automatisch inkrementiert wird



und die nächste Speicherzelle adressiert. PZ 1 dient dagegen zum Speichern der Rücksprungadresse bei Unterprogrammrutinen.

UP-Aufruf: RUF | n

Im Programmzähler PZ 0 steht die augenblickliche Adresse des Hauptprogramms. Sie wird nach PZ 1 gerettet und um 1 erhöht. Damit steht im PZ 1 die Rücksprungadresse. Anschließend wird der Adreßteil des UP-Aufrufes in



den Programmzähler PZ 0 geladen und mit n das Programm fortgesetzt. n ist die erste Adresse des Unterprogramms.

Rücksprung: POP
Der Befehl POP holt die Anschlußadresse (n+1) des Hauptprogramms aus PZ 1 nach PZ 0 zurück.

Zum Beispiel
Im Speicher steht der Befehl RUF 100. PZ 1 übernimmt die HP-Adresse 10 und inkrementiert sie. Die UP-Adresse 100 wird in PZ 0 geladen. POP signalisiert das Ende des UPs. PZ 1 und PZ 0 tauschen ihren Inhalt. Der Inhalt des Statusregisters kann mit dem Unterprogramm verändert werden. Es verhält sich wie ein normales Programm.

(Fortsetzung folgt)

Computer-Club-Umfrage

Computer-Clubs fördern den Informations-, Erfahrungs- und Programmaustausch zwischen Gleichgesinnten. Die FUNKSCHAU stellt derzeit eine Liste der heute existenten Clubs im deutschsprachigen Raum zusammen – dazu brauchen wir Ihre Hilfe!

Wenn Sie einem firmenunabhängigen Computer-Club mit mindestens zehn Mitgliedern angehören, so schreiben Sie uns bitte. Was wir benötigen? Nur folgende Angaben:

1. Genaue Anschrift des Clubs bzw. des Verantwortlichen.
2. Ungefähre Mitgliederzahl.
3. Von den Mitgliedern vorwiegend verwendete Computertypen.
4. Aufnahmebedingungen, Mitgliedsgebühren.

Senden Sie diese Angaben bitte möglichst bald an die Mikrocomputer-Redaktion der FUNKSCHAU, Postfach 37 01 20, 8000 München 37. Sie helfen damit Ihrem Club, neue Mitglieder zu gewinnen und dadurch den Erfahrungsaustausch zu aktivieren.

Ursula und Lutz May

RAM-Box

Die „RAM-Box" ist ein Hilfsmittel zur Entwicklung von Programmen, die später als EPROM zur Verfügung stehen sollen und daher meist in einem Adressenbereich liegen, der nicht im Entwicklungssystem belegt ist. Sie ersetzt das EPROM während der Programmerstellung.

Der Sinn der Sache

Der Mikroprozessor gibt die Möglichkeit, mit geringstem Aufwand digitale Probleme zu lösen, wie zum Beispiel die Steuerung von Gegensprechanlagen, Fernsteuersysteme, Morsehilfen (Decoder und Generator), Hobby-Spiele u. v. m. Es genügen heute schon weniger als zehn Bauelemente, um die dazu notwendige funktionsfähig Rechnerschaltung aufzubauen. Zur Lösung der oben angebenen Probleme ändert sich die Schaltung (Hardware) nicht. Lediglich das Programm (Software) muß den Aufgaben entsprechend neu geschrieben werden. Dazu benötigt man ein Entwicklungssystem. Damit ist man auch in der Lage, das Programm zu testen und es beliebig oft zu ändern. Wenn das Programm alle ihm gestellten Bedingungen erfüllt und für fehlerfrei gehalten wird, muß es in die endgültige Adressenform gebracht werden.

Im Entwicklungssystem stehen zur Programmerstellung nur ganz bestimmte RAM-Bereiche zur Verfügung. Ebenso haben die zum Programmtest notwendigen Ein- und Ausgabebausteine vom Entwicklungssystem her fest zugeteilte Adressen, die sich mit größter Wahrscheinlichkeit nicht mit denen der späteren Hardware decken. Bei höherer Programmiersprachen ist dies kein Problem, wohl aber bei einfacheren Entwicklungssystemen. Dort muß nun mit größter Sorgfalt die soeben erstellte Software auf den richtigen Adressenstand gebracht werden.

In einem EPROM konserviert kann nun das neue Programm in der endgültigen Schaltung eingesetzt und getestet werden. In den meisten Fällen funktioniert aber ein Programmteil nicht, oder das System reagiert nicht auf die im EPROM abgespeicherten Rechneranweisungen. Das kann an einem Hard- ware- oder Softwarefehler liegen. Die Fehlerursache läßt sich, sofern vorhanden, mit einem Logikanalysator finden. Die Funktion und Arbeitsweise sowie der Bau eines Logikanalysators werden in einem späteren Aufsatz beschrieben.

Handelt es sich um einen Softwarefehler, dann muß das EPROM auf jeden Fall neu geladen werden. Unter Umständen sind mehrere Umprogrammierungen erforderlich, bis alle Fehler beseitigt sind. Um diesen Arbeitsprozeß zu vereinfachen, gibt es RAM-Bausteine, die – mit einem Akku versehen – in einem kleinen Gehäuse montiert sind und die Aufgabe eines EPROMs übernehmen können. Diese RAM-Box kann vom Entwicklungssystem oder von einem Programmiergerät mit der Software geladen werden. Nach dem Laden schließt man die RAM-Box über einen dem EPROM gleichen Stecker an das zu testende Hardwaresystem an.

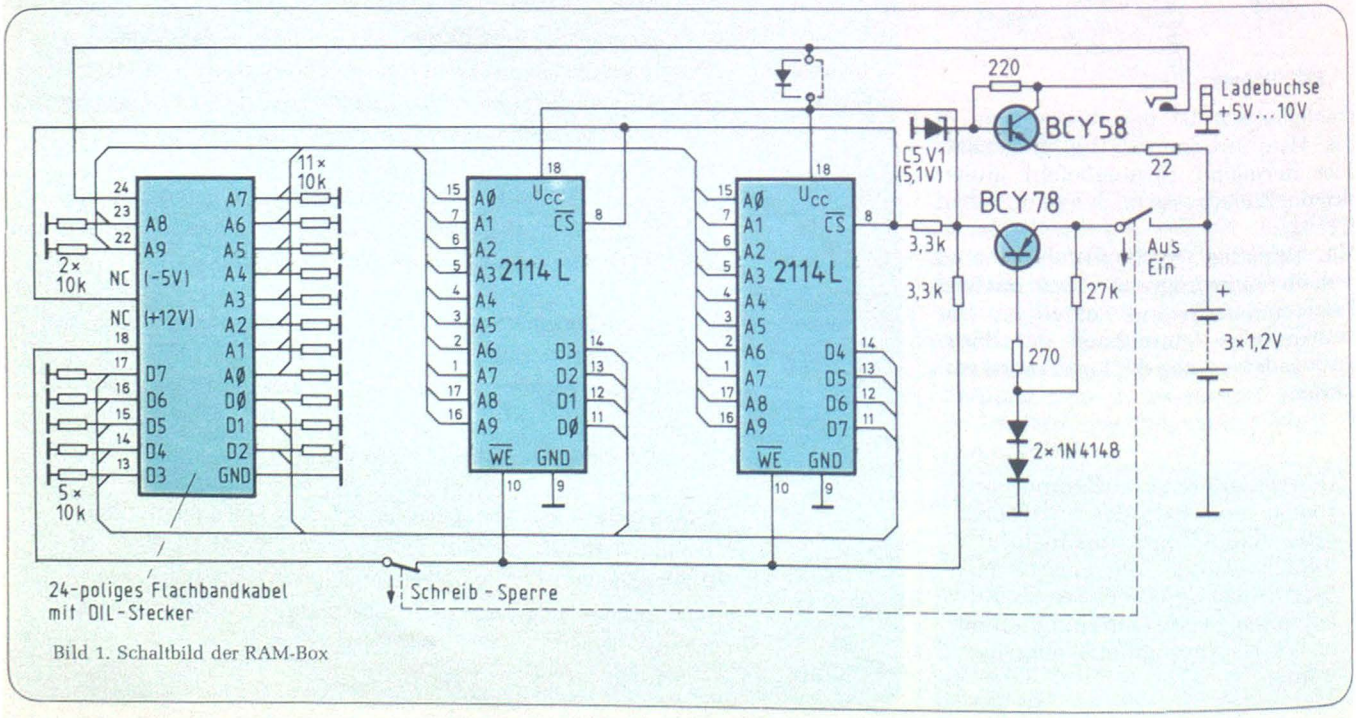


Bild 1. Schaltbild der RAM-Box

Notstromversorgung

Da die RAM-Box *(Bild 1)* mit Akkus ausgerüstet ist, kann die Information in den RAM-Bausteinen je nach Ausführung bis zu 4 Stunden ohne Versorgungshilfen von außen gehalten werden. Sobald die RAM-Box an ein System angeschlossen ist, bezieht sie von diesem ihren Arbeitsstrom; gleichzeitig werden die Akkuzellen geladen.

Da man die RAM-Bausteine im Betriebsfall von zwei Seiten versorgt (Akku und oder Systemnetz), kann es passieren, daß die Information in den RAM-Bausteinen verloren geht, sobald man das System ausschaltet. Da es aus Pegelgründen nicht möglich ist, mit einer Diode den RAM-Box-Akku vom Systemnetz zu trennen, würde nach dem Abschalten das zu testende Hardwaresystem von der RAM-Box versorgt. Dazu ist aber die Akkukapazität zu gering. Der Tiefentladeschutz der Box sorgt zwar dafür, daß der Akku durch diese Gewaltentladung keinen Schaden nimmt, aber die RAM-Information geht innerhalb weniger Minuten verloren.

Daher muß man, bevor das Hardwaresystem abgeschaltet wird, die RAM-Box abtrennen und, wenn nötig, die Ladebuchse an ein Netzteil mit + 5 V bis + 12 V anschließen. Trotz dieser Bedienungseinschränkung ist die RAM-Box ein sehr hilfreiches Gerät, um Mikroprozessorsysteme mit einem neuen Programm probeweise in Betrieb zu nehmen. *Bild 2* zeigt die Lötseite der Platine, *Bild 3* die Bauteileseite und *Bild 4* den Bestückungsplan.

Die Speicherkapazität ist 1K-8-bit und daher für den Simultaneinsatz des EPROM 2708 geeignet. Die Steckerbelegung ist so gehalten, daß sie nur + 5 V zur Versorgung benötigt und die eventuell vorliegenden + 12 V bzw. – 5 V nicht angeschlossen sind.

Die Steckerbelegung der RAM-Box *(Bild 5)* ist bis auf die Adressenleitung „A10" auch identisch mit dem EPROM-Typ 2716. Der EPROM-Baustein 2716 ist ein 2-K-Typ. Durch die fehlende Adressenleitung A10 taucht der Speicherinhalt zweimal im Adressenbereich auf. Werden nur kleinere Programme getestet, so spielt das jedoch keine Rolle.

Zusammenbau der Box

Unter Zuhilfenahme des Bestükkungsplans werden die Bauteile auf der Platine festgelötet. Links wird ein 2fach-Umschalter sowie eine kleine Miniatur-Klinkenbuchse mit Abschaltkontakt von der Bauteilseite her auf die Platine montiert. Die beiden Referenzspannungsdioden (1N 4148) für den Tiefentladeschutz werden in Brückenform zusammengelötet, da für sie nur zwei Lötaugen vorgesehen sind. Bis auf drei Widerstände sind alle Bauelemente stehend anzuordnen (siehe Bestückungsplan Nr. 1). Damit die 3 Akkus gut befestigt werden können, müssen zwei kleine Platinenstreifen aufrecht auf die dafür vorgesehenen Lötflächen angebracht werden (siehe Zeichnung). Die Versorgungsspannung wird von Kontakt 24 über eine kurze Drahtbrücke zu den ICs geleitet. Verwendet man CMOS-Speicherbausteine, kann sie gegen eine Germanium- oder Schottkydiode ausgetauscht werden.

Nach dem Bestücken müssen insgesamt 7 Drahtbrücken eingelötet werden. Die Akkus werden dann mit kurzen Drähten auf der Platine festgelötet *(Bild 6)*.

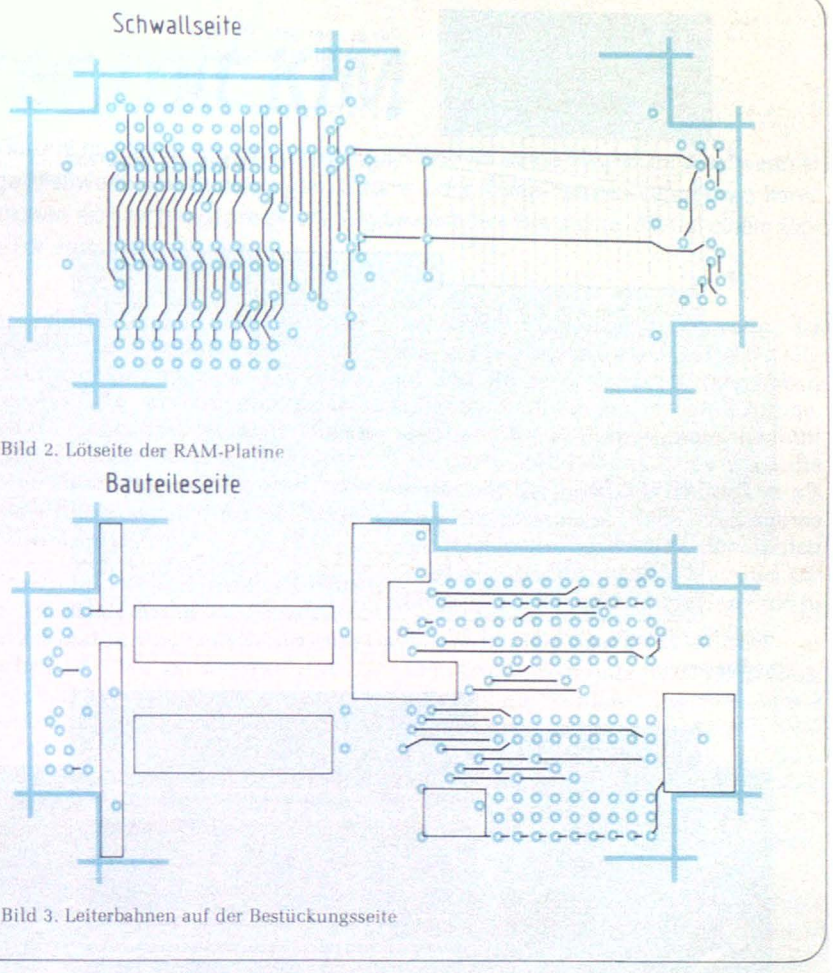


Bild 2. Lötseite der RAM-Platine

Bild 3. Leiterbahnen auf der Bestückungsseite

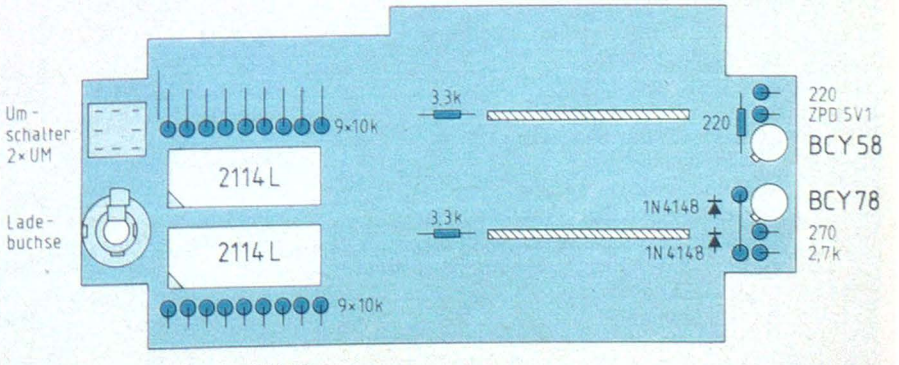


Bild 4. Bestückungsplan der RAM-Box

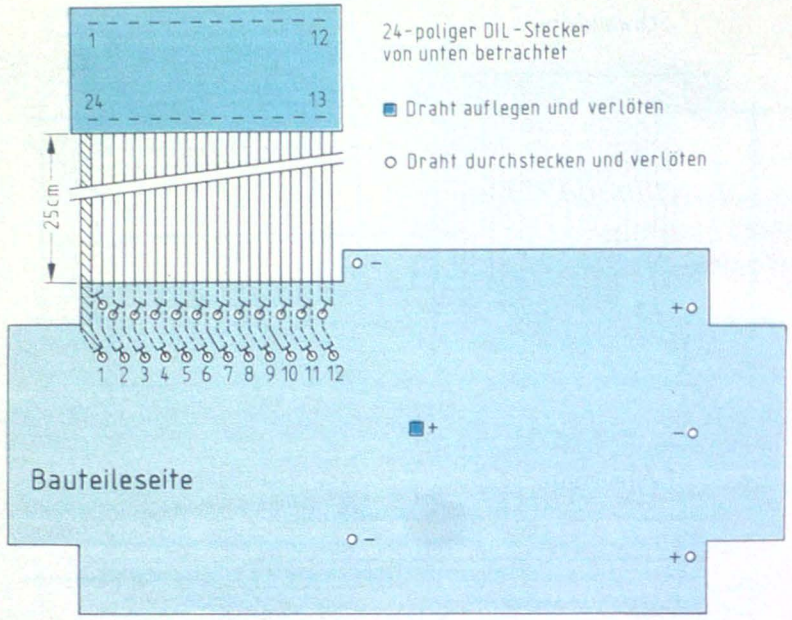


Bild 5. Die RAM-Platine wird über etwa 25 cm Flachbandleitung mit einem EPROM-Stecker verbunden

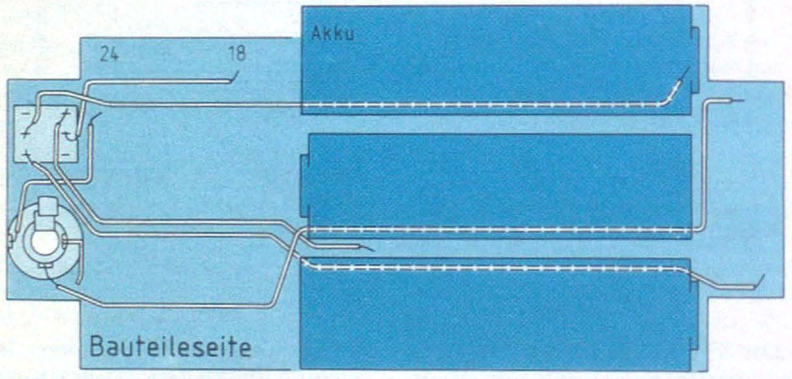


Bild 6. Verdrahtung der Akku-Zellen, des Umschalters und der Ladebuchse

Verwendete Bauteileliste

1	22 Ω ± 5 % RM 7,5	Bürklin
1	220 Ω ± 5 % RM 7,5	Bürklin
1	270 Ω ± 5 % RM 7,5	Bürklin
2	3,3 kΩ ± 5 % RM 7,5	Bürklin
18	10 kΩ ± 5 % RM 7,5	Bürklin
1	27 kΩ ± 5 % RM 7,5	Bürklin
2	1 N 4148 Si-Diode	SASCO
1	BCY 58 NPN-Transistor	SASCO
1	BCY 78 PNP-Transistor	SASCO
1	ZPD 5V1 Z-Diode	SASCO
2	18polige IC-Sockel	TISCO
2	2114 L 1Kx4bit RAM	National, Synertek, Itronik
1	Miniatur-Klinkenbuchse mit Abschaltkontakt	Bürklin Nr. 39 F 111
1	Miniatur 2fach Umschalter	
	28 cm Flachbandkabel (min 24polig)	Bürklin Nr. 10 G 304
1	24poliger DIL Flachbandkabelstecker	BERG Nr. 75220–112
2	Platinen 25x12 mm² zweiseitig beschichtet	
1	bopla-Gehäuse Nr. 410	Bopla, 4980 Bünde 1
1	RAM-Box-Platine	May-Elektronik, 8018 Grafing, Schlosserbreite 6

Schließlich wird das 24polige Flachbandkabel an der Platinenunterseite in die dafür vorgesehenen Lötaugen befestigt. Die Flachbandleitung sollte nicht länger als 25 cm sein (Gefahr von Signalübersprechen). Da der ganze Aufbau sehr klein gehalten wurde, sollte sauber und sorgsam gearbeitet werden.

Die Platine ist für den Einbau in ein „bopla"-Gehäuse vorbereitet (bopla Nr. 410: 100 x 50 x 25 mm³). Im Gehäuseboden sind die Platinenbefestigungen herauszubohren, da sie nicht gebraucht werden und Platz weg nehmen. Zwei Bohrungen sind für die Ladebuchse und den Schalter im Gehäuseboden anzubringen. Das Flachbandkabel ist durch einen seitlich zu feilenden Schlitz zu führen. Nach dem ersten elektrischen Test kann die RAM-Box zusammengeschraubt werden.

Ansteuerung der Error-LED beim Z-80-Kit

Gelegentlich wünscht man sich die Möglichkeit, die Fehleranzeige-LED am Z-80-Kit mit einem Befehl ansteuern zu können. Die LED hat die IO-Adresse 1F und wird über das Bit 7 des Datenbus gesteuert. Wenn Bit 7 gesetzt ist und die Adresse 1F ansteht (die oberen acht Adreßbits sind nicht von Bedeutung), leuchtet die LED, weil das zugehörige D-Flipflop (IC 6; 74LS74) gesetzt wird. Man setzt das Flipflop also mit den Befehlen:

```
3C50  3E80  LD A,80
3C52  D31F  OUT (1F) ,A
```

Man kann das D-Flipflop auch wieder rücksetzen, wenn man 00 auf den Datenbus legt und die IO-Adresse 1F aussendet. Das Ausschalten könnte zum Beispiel mit der Befehlssequenz erfolgen:

```
3C50  3E00  LD A,00
3C52  D31F  OUT (1F) ,A
```

Man kann auch ein einfaches Programm konstruieren, das die Error-LED blinken läßt. Hier folgt ein solches Programm. Das hier verwendete Unterprogramm „ZEITKO" ist ein Unterprogramm des Betriebssystemes, das den Prozessor ca. 0,7 s warten läßt, bevor die nächste Operation beginnt. Es gibt zwei verschiedene KIT-Betriebssysteme. Diese Version gilt für das erste Betriebssystem (Rev A – Rev C).

```
3C50  3E80     LD A,80
3C52  D31F     OUT (1F) ,A
3C54  CD6403   CALL ZEITKO
3C57  3E00     LD A,00
3C59  D31F     OUT (1F) ,A
3C5B  CD6403   CALL ZEITKO
3C5E  C3503C   JR 3C50
```

Michael Klein

Dipl.-Ing. Ekkehard Flögel

Digitalvoltmeter mit KIM

Eine Vielzahl nützlicher Anwendungen ergibt sich für einen Mikrocomputer, wenn er analoge Meßwerte wie Spannungen, Ströme oder Temperaturen verarbeiten kann. Dazu eignen sich recht gut preiswerte Digitalvoltmeter-Bausteine, die mit einem Eingangs-Tor verbunden werden.

Der CA 3162 E von RCA ist ein preiswerter Digital-Analog-Wandler, der in Verbindung mit dem Decoder-Treiber und drei 7-Segment-Anzeigen den Aufbau eines einfachen Digitalvoltmeters ermöglicht. Er kann aber auch allein in Verbindung mit einem Mikroprozessor zur Spannungsüberwachung oder zum Abtasten langsamer Vorgänge verwendet werden. Die Meßhäufigkeit beträgt wahlweise 4 oder 96 Messungen pro Sekunde. Dies ist, verglichen mit anderen A/D-Wandlern, relativ langsam, aber dem steht eine recht brauchbare Auflösung von –99 mV bis +999 mV gegenüber.

Die Ausgabe der Zahlenwerte erfolgt im Multiplex-Betrieb, d. h. die gewandelte Zahl wird über vier Leitungen im BCD-Code ausgegeben, während drei Leitungen angeben, um welche Dezimalstelle es sich handelt. Dadurch ist es sehr einfach, den Baustein an einen Mikroprozessor anzuschließen. In *Bild 1* ist nur die Grundschaltung angegeben. Die Beschaltung des Analogeingangs richtet sich nach der vorgegebenen Meßaufgabe. Die Ausgänge des Wandlers sind Stromausgänge, so daß diese bei Verwendung als Spannungsausgang mit einem Widerstand abgeschlossen werden müssen. Die Ausgänge für die Stellenanzeige sind „active low", die Ausgänge für die Dezimalstellen „active high".

Das Programm (*Bild 2*) besitzt 2 Warteschleifen. In der ersten wird abgewartet, ob die betreffende Stellenleitung Null wird. Ist die Bedingung erfüllt, wird nach einer kurzen Verzögerung die Dezimalstelle übernommen und durch das Hauptprogramm abgespeichert. Als Stellenzeiger im Mikroprozessor werden die Bits 0...2 der Speicherzelle 0003 verwendet. Durch Abfragen des Tores A und Maskieren mit dieser Zelle kann festgestellt werden, ob die gewünschte Dezimalstelle anliegt.

In der zweiten Warteschleife wird nach der Abspeicherung auf das Rücksetzen des Stellenzeigers gewartet. Ist dies erfolgt, so wird der Inhalt der Zelle 03 eine Stelle nach rechts verschoben und somit auf die nächste Stelle umgeschaltet. Das Hauptprogramm beginnt mit 04 in Zelle 03, d. h. es wird die Stelle mit der höchsten Wertigkeit als erste übernommen. Nach Übernahme der letzten Stelle werden die Zahlen passend zusammengeschoben und addiert, so daß das endgültige Ergebnis in den Zellen 01 und 00 abgelegt ist.

Ein Unterprogramm (0357) gibt diese Zellen ohne Formatierung fortlaufend auf einen Bildschirm aus. Dieses Programmstück kann auch durch ein Ausgabeprogramm ersetzt werden, das diese Zahlenwerte auf die Anzeige des KIM-1 ausgibt oder sie zur Weiterverarbeitung im Speicher ablegt.

Bei der diesem Programm zugrundegelegten Meßaufgabe werden nur positive Spannungswerte erfaßt. Deshalb wurde auf eine Auswertung von negativen Zahlenwerten verzichtet.

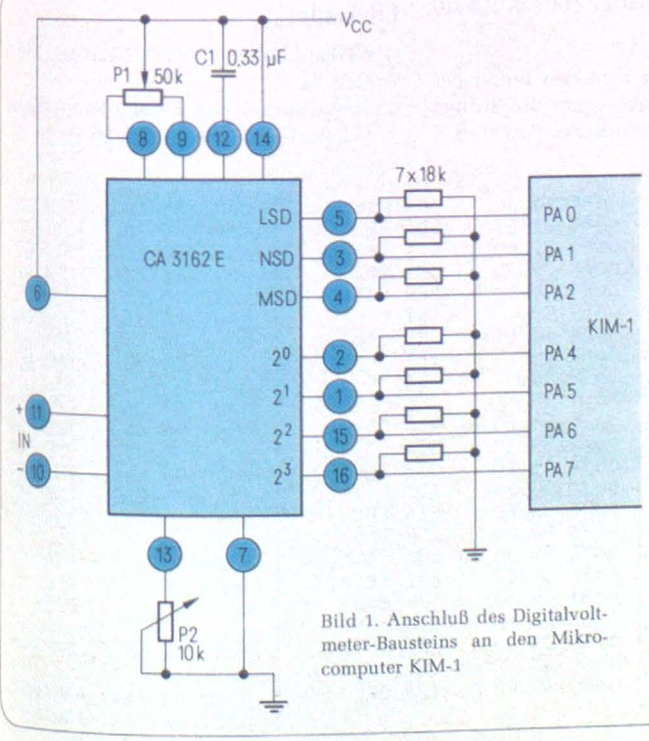


Bild 1. Anschluß des Digitalvoltmeter-Bausteins an den Mikrocomputer KIM-1

```
0300 A9 04     LDA #04
0302 85 03     STA 03
0304 20 36 03  JSR 0336
0307 85 00     STA 00
0309 20 48 03  JSR 0348
030C 20 36 03  JSR 0336
030F 85 01     STA 01
0311 20 48 03  JSR 0348
0314 20 36 03  JSR 0336
0317 85 02     STA 02
0319 46 02     LSR 02
031B 46 02     LSR 02
031D 46 02     LSR 02
031F 46 02     LSR 02
0321 A5 02     LDA 02
0323 18        CLC
0324 65 01     ADC 01
0326 85 01     STA 01
0328 46 00     LSR 00
032A 46 00     LSR 00
032C 46 00     LSR 00
032E 46 00     LSR 00
0330 20 57 03  JSR 0357
0333 4C 00 03  JMP 0300
```

```
0336 AD 00 17  LDA 1700
0339 25 03     AND 03
033B D0 F9     BNE 0336
033D A2 10     LDX #10
033F CA        DEX
0340 D0 FD     BNE 033F
0342 AD 00 17  LDA 1700
0345 29 F0     AND #F0
0347 60        RTS
0348 AD 00 17  LDA 1700
034B 25 03     AND 03
034D F0 F9     BEQ 0348
034F A2 10     LDX #10
0351 CA        DEX
0352 D0 FD     BNE 0351
0354 46 03     LSR 03
0356 60        RTS
0357 A9 0D     LDA #0D
0359 20 A0 1E  JSR 1EA0
035C A5 00     LDA 00
035E 20 3B 1E  JSR 1E3B
0361 A5 01     LDA 01
0363 20 3B 1E  JSR 1E3B
0366 4C 5A 1E  JMP 1E5A
```

Bild 2. Programm zur Ausgabe der Spannungswerte auf einem ASCII-Terminal. Nach jedem Meßwertausdruck wartet das Programm, bis der Benutzer eine Taste drückt. Will man die Meßwerte pausenlos ausdrucken, so ist an die Adresse 0366 der Befehl RTS (60) zu schreiben

Von A bis Z

Alphabetisches Sortieren mit dem 6502

Eine Kartei ist eine Datensammlung in meist alphabetischer Anordnung. Eine solche Anordnung ist zwar bei einer Computerkartei [1] zum Finden eines Stichwortes nicht mehr von so ausschlaggebender Bedeutung wie bei einer auf Karten vorliegenden, sie kann aber nützlich sein, wenn man eine Kartei durchmustern oder ausdrucken lassen möchte [2]. Das folgende Programm ordnet eine Kartei, die z. B. mit Hilfe des Programms aus Heft 24/78 erstellt wurde, in alphabetischer Reihenfolge. Im Gegensatz zu ähnlichen Programmen in BASIC benötigt dieses Maschinenprogramm z. B. für einen 4-KByte-Block nur wenige Sekunden.

Arbeitsweise des Programms

Das Programm vergleicht, am Textanfang beginnend, aufeinanderfolgende Zeilen miteinander. Als Zeilen werden hier bis zu 256 Zeichen lange Strings verstanden, die mit CR-Zeichen abgeschlossen sind. Stößt es auf zwei Zeilen, die in der falschen Reihenfolge stehen, so tauscht es die beiden aus: Zeile 1 wird in einen Pufferspeicher geladen, Zeile 2 auf den Platz von Zeile 1 verschoben und Zeile 1 hinter die neue Zeile 1 gehängt. Als Abschluß der Austauschaktion wird ein Flag gesetzt. Nun wird die neue Zeile 2 mit der nächsten Zeile verglichen, gegebenenfalls ausgetauscht usw.

Das Verfahren wird wiederholt, bis der Rechner das Textende erkennt (hex 3 E). Hier wird das Austauschflag abgefragt. Wenn im ganzen Durchgang, vom Anfang bis zum Ende des Textes, keine Austauschoperation vorgenommen wurde, ist das Programm beendet – alle Zeilen stehen in der richtigen Reihenfolge und es erfolgt ein Sprung zum KIM-Monitor[1]). Andernfalls springt das Programm an seinen Anfang zurück (0000).

Zeilenvergleich

Es werden nur Buchstaben und Ziffern zum Vergleich der Zeilen benützt, alle anderen Zeichen werden überlesen. Dazu dient ein Unterprogramm an der Adresse 0098. So wird beim Vergleich der Zeilen

PREIS...DM 4.70
PREIS...DM 42.00

die zweite Zeile an den Anfang gestellt, da für den Rechner die Zeilen bis zur 4 identisch sind, danach die 2 aber kleiner ist als die 7. Sind zwei unterschiedlich lange Zeilen bis zum Ende der kürzeren identisch, so wird die kürzere jeweils vorangestellt; Vertauscht würden z. B.

FUNKSCHAU-SONDERHEFT
FUNKSCHAU

Zeilen mit Ziffern am Anfang werden nach den Ziffern sortiert und den mit Buchstaben beginnenden Zeilen vorangestellt. Wie in [1] geht das Programm davon aus, daß der Text keine Kleinbuchstaben enthält; will man dies ändern, so ist der Inhalt der Zelle 00A6 auf 7B abzuändern.

[1]) Beim AIM-65 sollte man den Buffer bei 0170 beginnen lassen, wenn die Strings nicht länger als etwa 100 Zeichen sind.

Speicherbedarf

Das Programm liegt im Speicherbereich 0000 bis 00AF. In Seite Null werden die Zellen E0...E8 als Arbeitsspeicher benötigt. Außerdem ist ein Pufferspeicher erforderlich, der hier bei 0100 beginnt und dessen Umfang die maximale Zeilenlänge bestimmt (maximal 256 Zeichen). Das Programm prüft nicht nach, ob es den Pufferspeicher überschreibt, hier ist also Vorsicht geboten.

Zusätzlich sind noch zwei Speicherstellen mit der Anfangsadresse des Textbereiches belegt. Im Programm sind dies die Zero-Page-Zellen B0 und B1. Sie werden nur bei Adresse 0002 abgefragt und können bei Bedarf leicht durch Modifikation von Speicherzelle 0003 auf einen anderen Platz der Seite Null gelegt werden. Eine Änderung der Sortierrichtung ist zu erreichen, indem man bei Adresse 004E den Befehl BCS (hex B0) durch BCC (hex 90) ersetzt.

Das Programm läuft ohne Änderung auf dem KIM-1. Auf dem AIM-65 läßt es sich vorteilhaft zusammen mit den Editor-Befehlen des Monitorprogramms verwenden. Es ist darauf zu achten, daß der erste Text nicht mit CR beginnt und der letzte Text mit CR (hex 0D) und „>" (hex 3E) abgeschlossen ist. Die Textanfangsadresse in den Zellen 00B0 und 00B1 zeigt hier auf 0202, was dem Datensuche-Programm [1] entspricht.

Hans-Herbert Hagedorn

Literatur

[1] KIM auf Datensuche. FUNKSCHAU 1978, Heft 24

[2] Textausdruck mit Baudot-Fernschreiber. FUNKSCHAU 1979, Heft 16


```
0000 A2 02     LDX #02
0002 B5 AF     LDA AF,X
0004 95 E4     STA E4,X
0006 95 E6     STA E6,X
0008 CA        DEX
0009 D0 F7     BNE 0002
000B A0 00     LDY #00
000D 84 E2     STY E2
000F 84 E3     STY E3
0011 B1 E7     LDA (E7),Y
0013 E6 E7     INC E7
0015 D0 02     BNE 0019
0017 E6 E8     INC E8
0019 C9 0D     CMP #0D
001B D0 F4     BNE 0011
001D A4 E2     LDY E2
001F B1 E5     LDA (E5),Y
0021 C9 0D     CMP #0D
0023 F0 68     BEQ 008D
0025 20 98 00  JSR 0098
0028 90 F5     BCC 001F
002A 84 E2     STY E2
002C 85 E1     STA E1
002E A4 E3     LDY E3
0030 B1 E7     LDA (E7),Y
0032 C9 3E     CMP #3E
0034 D0 09     BNE 003F
0036 C5 E4     CMP E4
0038 F0 73     BEQ 00AD
003A 85 E4     STA E4
003C 4C 00 00  JMP 0000
003F C9 0D     CMP #0D
0041 F0 0D     BEQ 0050
0043 20 98 00  JSR 0098
0046 90 E8     BCC 0030
0048 84 E3     STY E3
004A C5 E1     CMP E1
004C F0 CF     BEQ 001D
004E B0 3D     BCS 008D
0050 A0 00     LDY #00
0052 B1 E5     LDA (E5),Y
0054 99 00 01  STA 0100,Y
0057 C9 0D     CMP #0D
0059 F0 04     BEQ 005F
005B C8        INY
005C 4C 52 00  JMP 0052
005F A0 00     LDY #00
0061 B1 E7     LDA (E7),Y
0063 91 E5     STA (E5),Y
0065 C9 0D     CMP #0D
0067 F0 04     BEQ 006D
0069 C8        INY
006A 4C 61 00  JMP 0061
006D C8        INY
006E 98        TYA
006F 18        CLC
0070 65 E5     ADC E5
0072 85 E7     STA E7
0074 A5 E6     LDA E6
0076 69 00     ADC #00
0078 85 E8     STA E8
007A A0 00     LDY #00
007C B9 00 01  LDA 0100,Y
007F 91 E7     STA (E7),Y
0081 C9 0D     CMP #0D
0083 F0 04     BEQ 0089
0085 C8        INY
0086 4C 7C 00  JMP 007C
0089 A9 FF     LDA #FF
008B 85 E4     STA E4
008D A5 E8     LDA E8
008F 85 E6     STA E6
0091 A5 E7     LDA E7
0093 85 E5     STA E5
0095 4C 0B 00  JMP 000B
0098 C8        INY
0099 C9 30     CMP #30
009B 90 0C     BCC 00A9
009D C9 3A     CMP #3A
009F 90 0A     BCC 00AB
00A1 C9 41     CMP #41
00A3 90 04     BCC 00A9
00A5 C9 5B     CMP #5B
00A7 90 02     BCC 00AB
00A9 18        CLC
00AA 60        RTS
00AB 38        SEC
00AC 60        RTS

00AD 4C 4F 1C  JMP KIM

00B0 02  TEXT-ANFANG L
00B1 02  TEXT-ANFANG H
```

6502-Programm zum Sortieren von Textblöcken nach dem Alphabet

Hans Werner

Mikrocomputer

Eine kleine Einführung

24. Teil

Ein komplettes Unterprogramm

Sucht ein Amateurfunker telegrafisch Kontakt mit außereuropäischen Stationen, sendet er den allgemeinen Anruf:

CQ DX CQ DX CQ DX de DL9ZZ usw. DL9ZZ ist sein persönliches Rufzeichen. Der Anruf wird solange wiederholt, bis eine Überseeverbindung hergestellt ist. Das kann unter Umständen recht lange dauern. Unser Funkamateur ist ziemlich faul und möchte seine Hand nicht unnötig an der Morsetaste überanstrengen. Er schreibt lieber ein Programm für seinen Mikrocomputer.

Ein Hauptprogramm ruft für jedes Morsesignal ein Unterprogramm auf, in dem die Zeiten für Punkt und Strich durch entsprechend viele Programmdurchläufe realisiert werden.

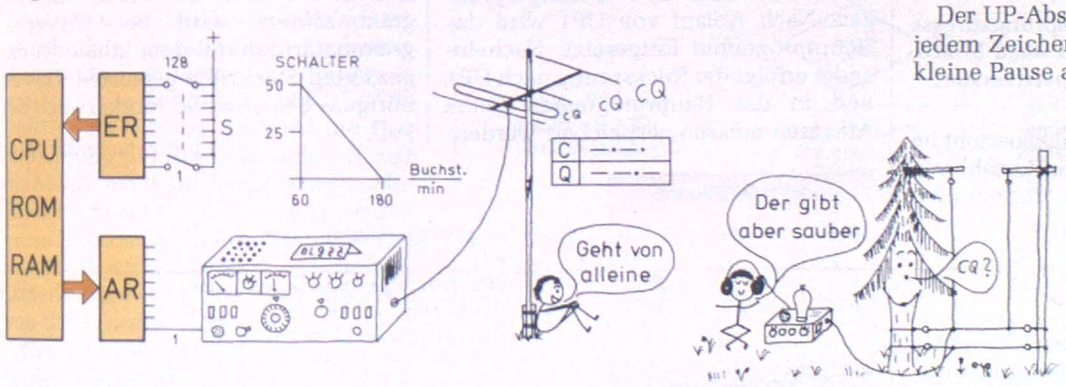
Das Verhältnis von Punkt zu Strich beträgt 1:3. Die kleine Pause zwischen den Zeichen wird auf die Länge eines Punktes, die große Pause zwischen den Buchstaben auf die Länge eines Striches festgelegt.

Mit den Schaltern S kann die Zeit für den Punkt vorgewählt werden. Im Unterprogramm ist eine Grundschleife von 200 vorhanden, so daß die Schalterstellung 50 in Wirklichkeit 10 000 Durchläufe bedeutet.

HP: Eingabe der Schalterstellungen und Errechnen der Durchläufe für den Strich. Jedes Zeichen benötigt 3 Befehle, von denen der letzte der UP-Aufruf TON bzw. PAUSE ist.

UP: Das UP gibt entweder für die Dauer eines Striches oder Punktes eine 1 in das Ausgaberegister AR. AR hält die 1 so lange für den Sender bereit, bis eine 0 eingegeben wird.

Der UP-Abschnitt PAUSE wird nach jedem Zeichen durchlaufen, damit die kleine Pause automatisch entsteht. Bei der großen Pause wird nur der untere Teil PAUSE abgearbeitet, allerdings mit mehr Durchläufen.



HAUPTPROGRAMM

	ORG		OD	
ANFANG	LA AC	d	512D	Eingabe: Punkt
	SP AC	d	PUNKT	Speichern.
	AD	d	PUNKT	Errechnen der
	AD	d	PUNKT	Durchläufe für
	SP AC	d	STRICH	den Strich.
	LA AC	d	STRICH	Ausgabe eines
	SP AC	d	ZEICHEN	Striches durch
	RUF		TON	UP-Aufruf.
	LA AC	d	PUNKT	Ausgabe eines
	SP AC	d	ZEICHEN	Punktes durch
	RUF		TON	UP-Aufruf.
	LA AC	d	STRICH	Ausgabe einer
	SP AC	d	ZEICHEN	Pause durch
	RUF		PAUSE	UP-Aufruf.
	JP	d	ANFANG	Rücksprung

UNTERPROGRAMM

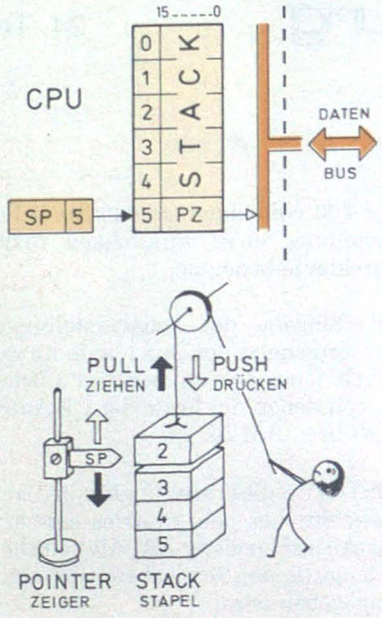
TON	LA AC	k	1D	
	SP AC	d	512D	Ton: 1 ausgeben
ZEIT 1	LA AC	k	200D	2 Programmschleifen sind
	SP AC	d	A	ineinander verschachtelt.
ZEIT 2	IN	d	A	Es ergeben sich:
	BR EX	d	ZEIT 2	ZEICHEN·200 Durchläufe.
	IN	d	ZEICHEN	Die Speicherzelle ZEICHEN
	BR EX	d	ZEIT 1	wird entweder mit STRICH
				oder PUNKT geladen.
	LA AC	d	PUNKT	Vorbereitung für die
	SP AC	d	ZEICHEN	kleine Pause.
PAUSE	LA AC	k	0D	
	SP AC	d	512D	Pause: 0 ausgeben
TIME 1	LA AC	k	200D	kleine Pause:
	SP AC	d	A	PUNKT · 200 Durchläufe
TIME 2	IN	d	A	
	BR EX	d	TIME 2	große Pause:
	IN	d	ZEICHEN	STRICH·200 Durchläufe
	BR EX	d	TIME 1	
	POP			Rücksprung in das HP.

VARIABLENLISTE

PUNKT	VARI	300D	Reservieren der Speicher-
STRICH	VARI	301D	plätze im RAM.
ZEICHEN	VARI	302D	(Adresse größer 256)
A	VARI	303D	

Der Stack

Die Organisation von Unterprogrammen ist bei den einzelnen Mikrocomputer-Typen verschieden. Verschachtelte Unterprogramme können mit unserem Modell nur mit viel Programmieraufwand realisiert werden,



denn es ist nur eine Rücksprungadresse möglich. Geeigneter sind dazu Mikrocomputer mit einem Stapelspeicher.

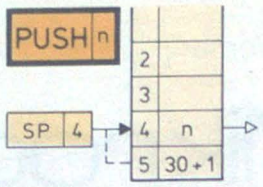
Ein Stapelspeicher (Stack) besteht im einfachsten Fall aus einer Anzahl von Registern, in denen die Rücksprungadressen gespeichert werden.

Beispiel:

5 Register bilden in der CPU einen Stack. Ist noch kein Unterprogrammsprung erfolgt, arbeitet das untere Register als Programmzähler. Ein zusätzlicher Zähler – der sog. Stackpointer – zeigt dann auf Register 5. Gleichzeitig schaltet er den Ein-Ausgabebus auf dieses Register. Im Unterschied zum Modell tauschen die Register nicht ihren Inhalt, sondern arbeiten nacheinander als Programmzähler. Mit 2 Befehlen sind auch verschachtelte Unterprogramm-Routinen möglich. Als Operationszeichen haben sich die Begriffe PUSH und PULL eingebürgert.

PUSH-Operation

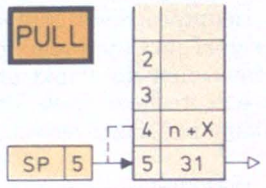
Register 5 arbeitet als Programmzähler PZ, bis z. B. mit Schritt 30 der Befehl PUSH n auftritt. Der Stackpointer wird von 5 auf 4 dekrementiert und die Adresse n in Register 4 geladen. Jetzt ist



Register 4 der PZ. Der Inhalt von 5 wird selbständig um 1 erhöht.

PULL-Operation

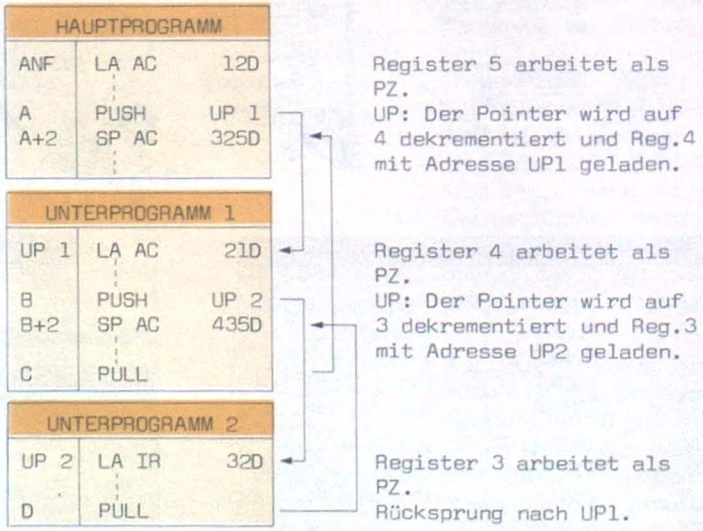
Register 4 arbeitet bis zum Befehl PULL als PZ. Dann wird der Stackpointer auf 5 inkrementiert und Register 5



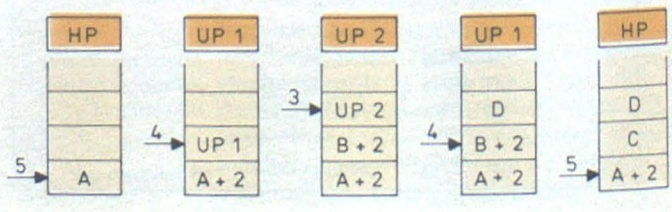
zählt mit der Rücksprungadresse 31 weiter. Der Inhalt von 4 bleibt erhalten.

Verschachtelte Unterprogramme

Das Hauptprogramm ruft mit PUSH UP1 das erste Unterprogramm auf. Innerhalb von UP1 wird ein zweites Unterprogramm UP2 benutzt, abgearbeitet und wieder nach UP1 zurückgesprungen. Nach Ablauf von UP1 wird das Hauptprogramm fortgesetzt. Nacheinander erfolgt der Rücksprung nach UP1 und in das Hauptprogramm. Beide Adressen müssen gespeichert werden.



Der Befehl PUSH n hat eine Länge von 2 Byte. Er belegt z. B. die Plätze A und A+1. Damit ist die Rückkehradresse nach der automatischen Inkrementierung A+2 usw.



Noch eine Möglichkeit

Bei manchen Mikrocomputern ist das Stackregister nicht in der CPU angeordnet, sondern ein beliebiger Speicherabschnitt (im RAM) kann entsprechend organisiert werden. In der CPU befindet sich nur der Stackpointer bzw. das Stackregister.

Soll in einem Programm der Stack verwendet werden, muß vorher der Stackpointer auf irgendeinen Ausgangswert z. B. 65000, zeigen. Dazu dient ein spezieller Befehl.

Der Programmzähler ist natürlich keine Zelle im Stackregister. Er befindet sich in der CPU. Der Inhalt des Programmzählers wird bei Unterprogrammaufrufen mit dem Inhalt der angezeigten Stackzelle getauscht. Das ist übrigens bei unserem Modell auch der Fall.

(Fortsetzung folgt)

Rüdiger Ebert

Lernen mit KIM

Hier soll keineswegs eine Lernmaschine vorgestellt werden, sondern eine weitere Möglichkeit zum Arbeiten (Spielen) mit dem KIM-1-Mikrocomputer. Das Programm ermöglicht das automatische Einlesen von Texten, die auf Kassette gespeichert sind. Diese Texte können Fragen enthalten, die dann vom Benutzer zu beantworten sind.

Nach Start des Programms (*Bild 1*, Startadresse 0047) erscheint auf dem Monitor der erste Text. Enthält dieser Text eine Frage, so ist diese zu beantworten. Das Programm ist so ausgelegt, daß max. vier Antwort-Möglichkeiten vorgegeben werden können, wobei eine oder mehrere Antworten richtig sein können. Die Antworten sind von 1 bis gegebenenfalls 4 durchnumeriert. Nach dem Betätigen der „Return"-Taste überprüft der Mikrocomputer die Antwort auf ihre Richtigkeit. Ist die Frage richtig beantwortet, wird das Wort „RICHTIG" auf dem Monitor sichtbar. Der Monitor wird nach zwei Sekunden gelöscht und ein neuer Text (Frage) vom Kassettenrecorder eingelesen. Die Zeit bis zum Löschen des Monitors wird ausgenutzt, um den Antrieb des Kassettenrecorders „hochlaufen" zu lassen. Bei falscher Beantwortung der Frage erscheinen die Worte „FALSCH! WIEDERHOLEN". Nach 2 s wird die Fehlermeldung gelöscht; danach hat man die Möglichkeit, die Frage erneut zu beantworten. Enthält der Text keine Frage, so ist nach Durcharbeiten des Textes nur die Taste „RETURN" zu betätigen. Danach wird ebenfalls ein neuer Text von der Kassette eingelesen.

Um die Wiedergabe des Kassettenrecordes durch Software steuern zu können, ist ein einfacher „Tape-Controller" notwendig. *Bild 2* zeigt die Schaltung; der KIM-Ausgang PA7 steuert über einen Transistor ein Relais. Der Arbeitskontakt des Relais ist mit der Start/Stop-Schaltung des Recorders zu verbinden. Ist eine solche Möglichkeit nicht vorhanden, so ist mit dem Relaiskontakt die Motorleitung zu unterbrechen.

Das in der KIM-Monitor-Software vorhandene Programm zum Einlesen der Daten von Kassette erlaubt leider keinen unmittelbaren Weiterlauf des Programms nach Beendigung des Lesevorganges. Das Programm springt nach Beendigung zur KIM-Startroutine und wartet auf eine Eingabe. Deshalb wurde für das hier vorliegende Programm eine modifizierte Version des Originalprogramms in die Speicherzellen 0100...01B9 aufgenommen.

Nach fehlerfreiem Einlesen erfolgt ein Sprung zur Adresse 0000; der Antrieb des Recorders wird gestoppt, und ein neuer Text erscheint auf dem Monitor. Wird beim Lesen der Kassette ein Fehler festgestellt, so erfolgt keine Feh-

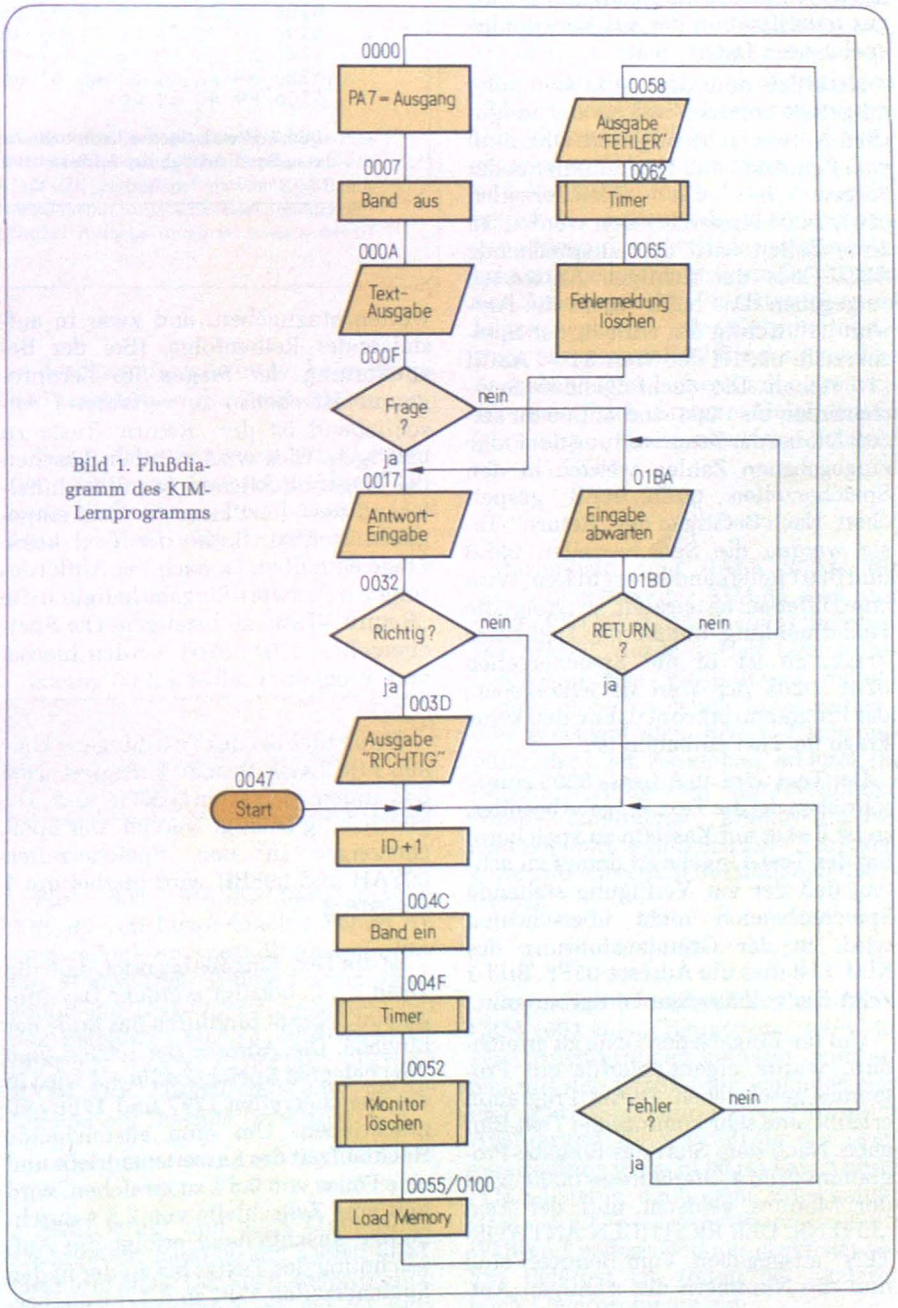


Bild 1. Flußdiagramm des KIM-Lernprogramms

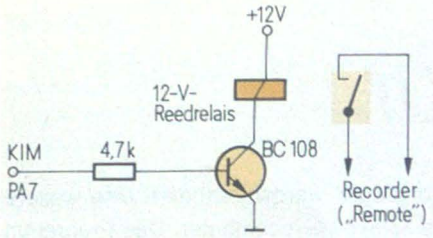


Bild 2. Mit einem Transistor und einem Reed-Relais läßt sich der Kassettenrecorder per Software fernsteuern

```
0000 A9 80 8D 01 17 A9 00 8D 00 17 20 AC 00 A9 00 CD
0010 01 02 D0 03 4C BA 01 A2 01 20 5A 1E C9 0D F0 09
0020 95 EA E8 20 9E 1E 4C 19 00 A9 0A 20 A0 1E A0 04
0030 A2 04 B9 00 02 D5 EA D0 1F CA 88 D0 F5 A2 0A B5
0040 C8 20 A0 1E CA D0 F8 EE F9 17 A9 80 8D 00 17 20
0050 96 00 20 A6 00 4C 00 01 A2 15 B5 D2 20 A0 1E CA
0060 D0 F8 20 96 00 A9 0D 20 A0 1E A2 14 20 9E 1E CA
0070 D0 FA A9 0B 20 A0 1E A9 0D 20 A0 1E A2 08 A9 09
0080 20 A0 1E CA D0 F8 20 8C 00 4C 17 00 A2 04 A9 00
0090 95 EA CA D0 F9 60 A2 08 A9 FF 8D 07 17 2C 07 17
00A0 10 FB CA D0 F3 60 A9 07 20 A0 1E 60 A9 05 85 FA
00B0 A9 02 85 FB A0 00 B1 FA C9 3B F0 09 20 A0 1E 20
00C0 63 1F 4C B4 00 20 8C 00 60 21 47 49 54 48 43 49
00D0 52 0A 0D 4E 45 4C 4F 48 52 45 44 45 49 57 20 21
00E0 48 43 53 4C 41 46 0A 0D
0100 A9 8D 8D EC 17 20 32 19 A9 4C 8D EF 17 A9 9B 8D
0110 F0 17 A9 01 EA 8D F1 17 A9 07 8D 42 17 A9 FF 8D
0120 E9 17 20 41 1A 4E E9 17 0D E9 17 8D E9 17 AD E9
0130 17 C9 16 D0 ED A2 0A 20 24 1A C9 16 D0 DF CA D0
0140 F6 20 24 1A C9 2A F0 06 C9 16 D0 D1 F0 F3 20 F3
0150 19 CD F9 17 F0 0D AD F9 17 C9 00 F0 06 C9 FF F0
0160 17 D0 54 20 F3 19 20 4C 19 8D ED 17 20 F3 19 20
0170 4C 19 8D EE 17 4C 84 01 20 F3 19 20 4C 19 20 F3
0180 19 20 4C 19 A2 02 20 24 1A C9 2F F0 14 20 00 1A
0190 D0 22 CA D0 F1 20 4C 19 4C EC 17 20 EA 19 4C 84
01A0 01 20 F3 19 CD E7 17 D0 0B 20 F3 19 CD E8 17 D0
01B0 03 4C 00 00 4C 47 00 4C 00 01 20 5A 1E C9 0D D0
01C0 F9 4C 47 00
```

Bild 3. Hex-Dump des Lernprogramms. Der Programmteil zum Lesen der Texte vom Band steht ab der Adresse 0100

lermeldung wie im KIM-1-Monitor, sondern ein Sprung zur Adresse 0047H; die ID-Nummer wird um 1 erhöht und der nächste Text eingelesen. Für „Nicht-KIM-Besitzer" sei erwähnt, daß die ID-Nummer eine Kennziffer ist, die zur Identifikation der auf Kassette gespeicherten Daten dient.

Natürlich muß dem Mikrocomputer mitgeteilt werden, welche der möglichen Antworten richtig sind. Dies muß vom Benutzer beim Programmieren der Fragen in die Speicherzellen 0201...0204 eingeschrieben werden. In diese Zellen wird der entsprechende ASCII-Code der richtigen Antworten eingegeben. Das heißt, wenn die Antwort „1" richtig ist, wird in der Speicherzelle 0201H der Wert 31 = ASCII „1" stehen. Die nachfolgenden Speicherzellen bis 0204 sind auf 00 zu setzen. Die bei der Beantwortung der Frage eingegebenen Zahlen werden in den Speicherzellen 00EB...00EB gespeichert. Nach Betätigen der „Return"-Taste werden die Speicherzellen 00EB und 0201 miteinander verglichen. Wird eine Differenz festgestellt, so erfolgt die Fehlermeldung. Enthält der Text keine Frage, so ist in die Speicherzellen 0201...0204 der Wert 00 einzugeben; das Programm erkennt daran, daß keine Frage im Text enthalten ist.

Der Text wird ab Adresse 0205 eingeschrieben. Ist die Text-Eingabe beendet, so ist dieser auf Kassette zu speichern. Bei der Text-Eingabe ist darauf zu achten, daß der zur Verfügung stehende Speicherbereich nicht überschritten wird. In der Grundausführung des KIM-1 ist dies die Adresse 03FF. *Bild 3* zeigt das vollständige Lernprogramm.

Um die Eingabe der Texte zu erleichtern, wurde eigens hierfür ein Programm geschrieben. Dieses Programm erlaubt eine sehr komfortable Text-Eingabe. Nach dem Start des Eingabe-Programms (*Bild 4*, Startadresse 0000) wird der Monitor gelöscht und der Text „LFD.NR. DER RICHTIGEN ANTWORTEN" ausgegeben. Vom Benutzer sind nun die Nummern der richtigen Antworten einzugeben, und zwar in aufsteigender Reihenfolge. (Bei der Beantwortung der Fragen im Lernprogramm ist ebenso zu verfahren.) Anschließend ist die „Return"-Taste zu betätigen. Dies wird mit dem Löschen des augenblicklichen Monitors-Inhaltes quittiert. Jetzt kann der Text eingegeben werden. Sollte der Text keine Frage enthalten, ist nach der Aufforderung zur Antwort-Eingabe lediglich die „Return"-Taste zu betätigen. Die Speicherzellen 0201...0204 werden hierbei vom Programm selbst auf 00 gesetzt.

Tippfehler bei der Text-Eingabe können mit „Back Space" korrigiert werden, indem die Tasten „CRTL" und „H" gleichzeitig betätigt werden. Der Speicherzeiger in den Speicherzellen 00FAH und 00FBH wird hierbei um 1 erniedrigt.

Ist die Text-Eingabe beendet, muß die Taste „ ; " betätigt werden. Das Programm erkennt hierdurch das Ende der Eingabe. Die Adresse der letzten vom Text belegten Speicherzelle + 1 wird in die Speicherzellen 17F7 und 17F8 eingeschrieben. Um eine ausreichende Hochlaufzeit des Kassettenantriebs und eine Pause von 0,5 s zu erreichen, wird nun eine Zeitschleife von 2,5 s durchlaufen; anschließend erfolgt die Aufzeichnung des Textes bis zu der in den Speicherzellen 17F7 und 17F8 angegebenen Endadresse. Die Steuerung des Kassettenrecorders erfolgt hierbei ebenfalls über den schon beschriebenen Tape-Controller. Nach Beendigung der Aufzeichnung kann unmittelbar der nächste Text eingegeben werden. Anzumerken ist noch, daß dem ersten Text die ID-Nummer 02 zugewiesen wird. Diese wird im weiteren Programmverlauf automatisch um jeweils 1 erhöht.

Um später beim Ablauf des Lern-Programms eine möglichst kurze Zugriffszeit zum Text zu erhalten, wird dieser nicht mit der entsprechenden KIM-Routine aufgezeichnet, sondern mit dem im „The first Book of KIM" beschriebenen Programm „Hypertape". Dieses Programm erlaubt eine etwa 6mal schnellere Aufzeichnung und ist zur vorhandenen KIM-Hardware voll kompatibel. Es ergibt sich hierbei eine durchschnittliche Zugriffszeit von 5 s.

Um einen störungsfreien Ablauf der beiden Programme zu gewährleisten, sollten nachfolgende Hinweise beachtet werden. Das Lern-Programm und das Eingabe-Programm müssen mit der ID-Nummer 01H aufgezeichnet werden. Soll der Text auf der gleichen Kassette aufgezeichnet werden wie das Lern-Programm, so ist zwischen beiden Aufzeichnungen eine Aufzeichnungs-Pause von 8...10 s einzuhalten. Dies ist notwendig, da der Recorder nach dem

```
0000 A9 80 8D 01 17 A9 00 8D 00 17 A2 04 A9 00 9D 00
0010 02 CA D0 F8 20 62 00 A2 23 B5 A4 20 A0 1E CA D0
0020 F8 EE F9 17 A2 01 20 5A 1E C9 0D F0 0A 9D 00 02
0030 E8 20 9E 1E 4C 26 00 20 68 00 20 62 00 20 78 00
0040 20 63 1F A5 FA 8D F7 17 A5 FB 8D F8 17 A9 00 8D
0050 F5 17 A9 02 8D F6 17 A9 80 8D 00 17 20 68 00 4C
0060 00 01 A9 07 20 A0 1E 60 A2 0A A9 FF 8D 07 17 2C
0070 07 17 10 FB CA D0 F3 60 A9 05 85 FA A9 02 EA EA
0080 85 FB 20 5A 1E A0 00 C9 08 F0 0C 91 FA C9 3B F0
0090 13 20 63 1F 4C 82 00 C6 FA A9 FF C5 FA D0 02 C6
00A0 FB 4C 82 00 60 3A 4E 45 54 52 4F 57 54 4E 41 20
00B0 4E 45 47 49 54 48 43 49 52 20 52 45 44 20 2E 52
00C0 4E 20 2E 44 46 4C 20

0100 A9 AD 8D EC 17 20 32 19 A9 27 85 F5 A9 BF 8D 43
0110 17 A2 64 A9 16 20 61 01 A9 2A 20 88 01 AD F9 17
0120 20 70 01 AD F5 17 20 6D 01 AD F6 17 20 6D 01 20
0130 EC 17 20 6D 01 20 EA 19 AD ED 17 CD F7 17 AD EE
0140 17 ED F8 17 90 E9 A9 2F 20 88 01 AD E7 17 20 70
0150 01 AD E8 17 20 70 01 A2 02 A9 04 20 61 01 4C C2
0160 01 86 F1 48 20 88 01 68 C6 F1 D0 F7 60 20 4C 19
0170 48 4A 4A 4A 4A 20 7D 01 68 20 7D 01 60 29 0F C9
0180 0A 18 30 02 69 07 69 30 A0 07 84 F2 A0 02 84 F3
0190 BE BE 01 48 2C 47 17 10 FB B9 BF 01 8D 44 17 A5
01A0 F5 49 80 8D 42 17 85 F5 CA D0 E9 68 C6 F3 F0 05
01B0 30 07 4A 90 DB A0 00 F0 D7 C6 F2 10 CF 60 02 C3
01C0 03 7E HYPERTAPE WURDE VON J.BUTTERFIELD ERSTELLT
01C2 20 8C 1E 4C 00 00
```

Bild 4. Programm zur Eingabe der Texte auf Band. Es enthält das bekannte Hypertape-Programm von Jim Butterfield (0100...01C1), wobei der Rücksprungbefehl an der Adresse 015E steht. Ersetzt man ihn durch 4C 25 19, so läßt sich „Hypertape" statt der KIM-Bandroutine (1800) verwenden; die Aufzeichnung geschieht mit sechsfacher Geschwindigkeit, ist aber mit der Leserroutine (1873) völlig kompatibel. Anfangs- und Endadresse sowie ID werden wie üblich gesetzt

Einlesen des Lern-Programms noch mit Hand gestoppt werden muß.

Zum Löschen des Monitor-Bildschirms wurde der Code 07 verwendet. Dieser muß u. U. in Abhängigkeit vom verwendeten Terminal geändert werden. Beim Lern-Programm steht dieser in der Speicherzelle 00A7, beim Eingabe-Programm in der Zelle 0063.

Wenn diese Hinweise beachtet werden dürfte der Programm-Ablauf keinerlei Schwierigkeiten bereiten. Und nun viel Spaß beim „Lernen mit KIM"!

µC-Nachrichten

Fairchild – kürzlich weitgehend an den Schlumberger-Konzern verkauft – brachte einen **Einplatinen-Computer** namens F 387X PEP heraus. Er arbeitet mit der **CPU 3850** und besitzt auf der Platine ein Hexadezimal-Tastenfeld sowie eine sechsstellige Anzeige. Das Konzept ähnelt sehr dem bekannten KIM-1, wenn man einmal von der Verwendung einer anderen Chipfamilie absieht. In der Grundausstattung besitzt das System 2 KByte ROM mit dem Monitorprogramm und 2 KByte RAM. Es gestattet die Programmentwicklung für die Mikrocomputer MK bzw. F 3870, 3872, 3876 und 3878.

Für den BASIC-Mikrocomputer **ABC-80** von Luxor brachte Sattco AB (Schweden) ein speziell zugeschnittenes **Floppy-Disk-System** heraus. Dazu gehört ein Disk-Operating-System mit den Befehlen CMDINT, SPACE, MAP, DOSGEN, COPY, COPYLIB, LIB und BASIC ERR. Zusätzlich ist ein Z-80-Assembler auf Diskette lieferbar.

□

AMD fertigt nun auch Einchip-Mikrocmputer, nämlich die Typen **8041, 8048 und 8035**. Der 8048, ursprünglich von Intel eingeführt, besitzt 64 Byte RAM, 1 KByte ROM, 27 I/0-Leitungen und eine CPU mit 2,5 µs Befehlszyklus. Auf dem Chip sind auch ein 8-bit-Timer/Zähler und der Taktoszillator.

□

SGS-Ates vereinbarte mit Zilog die Distributionsrechte für die Mikroprozessoren **Z-8, Z-80 und Z-8000** sowie für die dafür passenden Entwicklungssysteme. Außerdem erhielt SGS die alleinigen europäischen Herstellerrechte für die Prozessoren Z-8 und Z-8000.

□

Um Studenten und Ingenieuren die Möglichkeit zu geben, sich in Hardware- und Software-Probleme moderner Mikroprozessoren einzuarbeiten, gründete der VDE-Bezirksverein Braunschweig eine **Arbeitsgruppe „Mikrocomputer/Mikroprozessoren"**. Die Verantwortlichen des Braunschweiger VDE wollen damit die Aus- und Weiterbildung auf diesem Gebiet betreiben, andererseits aber auch eigene Ideen der Mitglieder in diese Arbeitsgruppe mit einbringen. Das Lernen und Arbeiten soll in Projektgruppen erfolgen, wobei in jeder Gruppe mindestens ein „Spezialist" für das jeweilige Aufgabengebiet sitzen soll. Kontaktadresse: Ulrich Knappe, Oelschlägern 21, 3300 Braunschweig, Telefon: 05 31/4 58 51.

□

Assembler- und BASIC-ROMs für den Mikrocomputer **AIM-65** sind nun von GWK (Tel. 02451/41911) lieferbar. Der AIM-65 wurde in Heft 10 ausführlich besprochen und stellt wohl derzeit – mit entsprechenden ROMs erweitert – den preiswertesten BASIC-Tischcomputer dar. Der Assembler arbeitet im Gegensatz zu dem im AIM-65-Monitor bereits enthaltenen Line-by-line-Assembler mit symbolischer Adressierung und entspricht der üblichen 6502-Syntax.

□

Der in Heft 7 vorgestellte „Profi-PET" **CBM 3001** von Commodore wird in zwei Ausführungen geliefert; zum einen mit den Grafik-Möglichkeiten des PET 2001 und zum anderen in einer Ausführung, die speziell für die Textverarbeitung gedacht ist und auch ohne POKE-Befehle nach dem Einschalten Groß- und Kleinschreibung zuläßt. Während erstere Version praktisch sofort lieferbar ist, treten bei letzterer noch Lieferprobleme auf.

Michael Klein

Software-Uhr für den Z-80-KIT

Unter einer Software-Uhr versteht man ein Programm, das durch geeignete Verzögerungsschleifen einen „Software-Zähler" so weiterstellt, daß bestimmte Speicherzellen die aktuelle Uhrzeit enthalten, die dann z. B. auf einer Siebensegment-Anzeige ausgegeben werden kann.

Das eigentliche Problem einer Software-Uhr liegt darin, daß sie in einer Endlosschleife läuft, in der es Verzweigungen gibt, aber dennoch nach einer Sekunde den Inhalt des Sekundenregisters erhöhen muß. Da durch die Sprünge das Programm einmal mehr und einmal weniger Schritte ausführen muß, gilt es die dadurch entstehende Zeitdifferenz durch geschickt angelegte Warteschleifen zu kompensieren. Aber auch hier gibt es Probleme, denn z. B. ein Sprungbefehl JR NZ e wird in 12 Taktzyklen ausgeführt, wenn die Sprungbedingung erfüllt ist; er braucht aber nur 7 Taktzyklen, wenn die Sprungbedingung nicht erfüllt ist. Das *Bild* zeigt ein Z-80-Programm. Allerdings wurde das Programm nicht so ausgelegt, daß die Schleife rechnerisch genau eine Sekunde für einen Durchlauf braucht. Es mußte auch die Zeitverzögerung mit einbezogen werden, die das WAIT-Flipflop verursacht. Dieses Wait-Flipflop ist auch für die Fehler verantwortlich, die bei der Zeitangabe auftreten. Deshalb wurde das Programm auch nicht auf 1 Mikrosekunde genau ausgetüftelt, obwohl dies möglich wäre. Das Programm enthält „Feh-

```
Sekundenspeicher:       3CDØ
Minutenspeicher:        3CD1
Stundenspeicher:        3CD2
Stackpointer:           3CFF
Programmzeiger:         3C5Ø

ANF: 3C5Ø   21   LD HL,3CDØ
       51   DØ
       52   3C
       53   ØE   LD C,6Ø
       54   6Ø
       55   CD   CALL LIMIT
       56   93
       57   3C
       58   2Ø   JR NZ DIS1
       59   14
       5A   36   LD (HL),ØØ
       5B   ØØ
       5C   23   INC HL
       5D   CD   CALL LIMIT
       5E   93
       5F   3C
       6Ø   2Ø   JR NZ DIS
       61   1Ø
       62   36   LD (HL),ØØ
       63   ØØ
       64   23   INC HL
       65   ØE   LD C,24
       66   24
       67   CD   CALL LIMIT
       68   93
       69   3C
       6A   2Ø   JR NZ DIS
       6B   Ø6
       6C   36   LD (HL),ØØ
       6D   ØØ
DIS1:3C6E   Ø6   LD B,Ø5
       6F   Ø5
A:     7Ø   1Ø   DJNZ A
       71   FE
DIS:   72   Ø6   LD B,6D
       73   6D
B:     74   1Ø   DJNZ B
       75   FE
       76   Ø6   LD B,Ø2
       77   Ø2
E:     78   C5   PUSH BC
       79   Ø6   LD B, B2
       7A   B2
       7B   C5   PUSH BC
D:     7C   21   LD HL,3CDØ
       7D   DØ
       7E   3C
       7F   FD   LD YI,ØØ11
       8Ø   21
       81   11
       82   ØØ
       83   Ø6   LD B,Ø3
       84   Ø3
C:     85   CD   CALL DISPL
       86   99
       87   3C
       88   1Ø   DJNZ C
       89   FB
       8A   C1   POP BC
       8B   1Ø   DJNZ D
       8C   EE
       8D   C1   POP BC
       8E   1Ø   DJNZ E
       8F   E8
       9O   C3   JP ANF
       91   5Ø
       92   3C
;
; Unterprogramm LIMIT
;
3C93   7E   LD A,(HL)
  94   3C   INC A
  95   27   DAA
  96   77   LD (HL),A
  97   91   SUB C
  98   C9   RET
;
; Unterprogramm DISPL2
;
3C99   7E   LD A,(HL)
  9A   CD   CALL ROUT
  9B   A6
  9C   3C
  9D   7E   LD A,(HL)
  9E   CD   CALL SRL 4
  9F   19
  AØ   Ø1
  A1   CD   CALL ROUT
  A2   A6
  A3   3C
  A4   23   INC HL
  A5   C9   RET
;
; Unterprogramm ROUT
;
3CA6   E6   AND ØF
  A7   ØF
  A8   DD   LD IX, Ø3DE
  A9   21
  AA   DE
  AB   Ø3
  AC   16   LD D,ØØ
  AD   ØØ
3CAE   5F   LD E,A
  AF   DD   ADD IX,DE
  BØ   19
  B1   DD   LD A,(IX+d)
  B2   7E
  B3   ØØ
  B4   FD   LD C,(IY)
  B5   4E
  B6   ØØ
  B7   ED   OUT (C),A
  B8   79
  B9   FD   INC IY
  BA   23
  BB   ØØ   NOP
  BC   C9   RET
```

Dieses Programm macht aus dem Z-80-KIT eine Digitaluhr

ler" von einigen hundert Mikrosekunden pro Durchlauf, die man jedoch mit einigem Zusatzaufwand loswerden könnte.

Das Programm ist so aufgebaut, daß es etwa 0,995 s lang die aktuelle Zeit anzeigt und während 0,005 s die neue Anzeige errechnet. Der Zeiger HL zeigt auf die Speicherzelle, deren Inhalt gerade behandelt wird. Das Einstellen von HL geschieht im ersten Programmschritt.

Dann wird der Grenzwert für die Sekundenanzeige gesetzt (60). Das Unterprogramm LIMIT erhöht den Inhalt der Speicherzelle, auf die HL zeigt und stellt fest, ob der Grenzwert 60 erreicht ist. Wenn nicht, verzweigt nach der Rückkehr aus dem Unterprogramm das Hauptprogramm nach der Abfrage JR NZ DIS1 nach rechts. Dann folgt eine Warteschleife. Sie bewirkt die gleiche Zeitverzögerung, wie wenn das Programm nach JR NZ DIS1 nach unten weiterverzweigt hätte. Das gleiche gilt auch für die beiden nachfolgenden Abfragen. Bei der letzten Abfrage, die nur jede Stunde erreicht wird, wurde die zusätzliche Warteschleife weggelassen. Die zweite Warteschleife mit dem Parameter B=6D dient zur Ergänzung des Programmes auf eine Sekunde „Zykluszeit". Nun folgt der Teil des Programmes, in dem die Anzeige erfolgt. Hier sind zwei Zählschleifen eingebaut.

```
LIST
 6 PRINT"[CLR][WHT]";TAB(6)"0  1  2  3  4  5
  6  7  8  9":FORQ=0TO39:PRINT"_";:NEXT:
PRINT
 7 PRINT"[RVS]   [OFF]";:FORQ=0TO15:PRINTTAB(5)"|"
:NEXT:GOSUB250:M=33094
 9 Z=X/40:S=X-AB:AB=0:POKE(M+40*Z+S),U:U=U+
1:X=X+1:H=H+1:M=M+1
 15 IFH=10THENM=M+10:P=P+H:H=0
 19 IFP=130THENGOSUB200
 40 PRINT"[HOME]";U;T=TI:IF(TI-T)<ABTHEN40
 45 IFU>255THEN70
 60 GOTO9
 70 PRINT"[RVS]                    [OFF]";TAB(22)
"                    "
 200 GETA$:IFA$=""THEN200
 220 GOSUB270:M=33094:X=0:P=0:RETURN
 250 FORQ=0TO13:PRINT"[DOWN]";:NEXT
 260 FORQ=0TO12:PRINTQ:NEXT:PRINT"[RVS]":RET
URN
 270 FORQ=0TO6:PRINT"[RVS]";:NEXT:FORQ=13TO2
5:PRINTQ:NEXT:RETURN
READY.
```

Bild 3. Programm zur Darstellung der Bildschirm-Zeichencodes

Die ganz innen liegende Schleife mit B=3 dient lediglich dazu, alle drei Zeitwerte zur Anzeige zu bringen. Die beiden äußeren Schleifen mit B=B2 und B=02 braucht man, um die Anzeigedauer von ca. 0,995 s zu erreichen.

DISPL2 lädt den anzuzeigenden Wert in A und übergibt ihn an das Unterprogramm ROUT, das die eigentliche Anzeigensteuerung übernimmt. DISPL2 trennt das anzuzeigende Byte in das obere und untere „Nibble" auf und übergibt es an ROUT. Je nachdem, welche Zeitkonstante der WAIT-Monoflop ausweist, kann eine Änderung der Konstanten 6D erforderlich werden.

Nachhilfe für den PET 2001

Die Hardware des PET 2001 ist – zumindest bei den ersten Serien – etwas verbesserungsbedürftig. Der nachfolgende Beitrag beschreibt einfache Erweiterungen, die den PET-Komfort deutlich steigern.

Spannungsversorgung

Die Spannungsversorgung des 8-KByte-PET bietet mehr Reserven als die im Handbuch angegebenen 300 mA. Allerdings ist dann ein zusätzlicher Spannungsregler erforderlich *(Bild 1)*, da zwar Trafo und Gleichrichter mehr Strom abgeben können, nicht aber der 5-V-Regler. Statt des 7805 kann natürlich auch ein anderer 5-V-Regler zum Einsatz kommen. Die Zusatzschaltung liefert max. etwa 1 A z. B. für periphere Geräte oder Speichererweiterungen. Der Aufbau erfolgt am besten auf einer kleinen Lochrasterplatte, die in ein Miniatur-Gehäuse gesetzt wird und unter dem PET-Tastenfeld Platz findet. Recht praktisch ist es, die neue Spannungsversorgung z. B. über Bananen- oder Eisenbahnbuchsen nach außen zu führen.

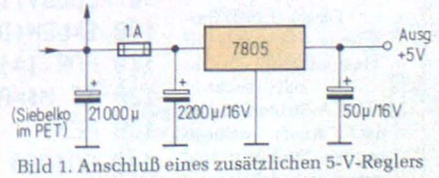


Bild 1. Anschluß eines zusätzlichen 5-V-Reglers an den PET. Mit ihm kann etwa 1 A entnommen werden

Tonerzeugung

Bild 2 zeigt eine einfache Schaltung zum Anschluß eines Lautsprechers an einem PET-Port. Die gleiche Schaltung eignet sich auch, um die Kassettenaufzeichnung über den READ-Anschluß des Kassettensteckers im PET mithören zu können. Die Kassettenaufzeichnung läßt sich auch am USER-Port abnehmen.

Der Nachleuchtfleck

Das lange Nachleuchten auf dem Schirm des PET nach dem Ausschalten ist nicht nur unschön, sondern kann auch zu Einbrennflecken auf der Bildröhre führen. Aus diesem Grunde sollte man den 3,3-µF/200-V-Kondensator an der vorderen Kante der Monitor-Platine direkt unter dem Bildröhrensockel gegen einen 50-µF-Typ austauschen, wobei eine Spannungsfestigkeit von 70 V genügt. Neuere PETs zeigen die Erscheinung des Nachleuchtens nicht mehr.

Kontaktprobleme

Die Verbindungen der Mutterplatine zum Tastenfeld, zum Monitor, zum

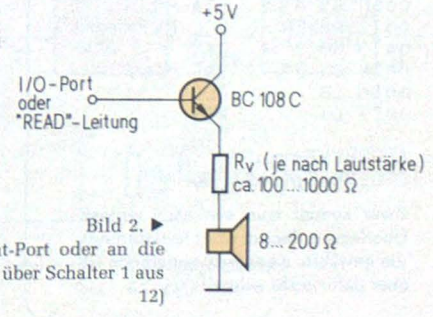


Bild 2. ▶
Anschluß eines Lautsprechers an einen Output-Port oder an die READ-Leitung des Kassettenrecorders (eventuell über Schalter 1 aus 12)

Kassettenrecorder und zur Stromversorgung sind als Steckkontakte ausgeführt und haben schon oft zu Problemen geführt. Die radikalste Lösung des Steckerproblems wäre ein Verlöten der Anschlüsse auf der Platine. Dies führt jedoch zu weiteren Problemen, weil dann beim Öffnen des PET u. U. die Verbindungen zu kurz werden.

Trotzdem ist diese Lösung zumindest für die Speisespannungsstecker dringend zu empfehlen: Beim Autor führte diese Verbindung wegen eines seitlichen Spiels von 2 mm zu einem Totalausfall des Geräts. Der Stecker wurde kurzerhand abgeschnitten (meist ist er an der Braunfärbung an Pin 2 gut zu erkennen); die Kontakte an der Steckleiste werden gegenseitig verbogen, so daß alle „+"-Anschlüsse in Richtung Trafo und alle „–"-Anschlüsse in Richtung Keyboard zeigen. Dann werden die Leitungen direkt verlötet. Alle übrigen Steckverbindungen sollte man von Zeit zu Zeit mit Kontaktspray o. ä. behandeln (Stecker und Stifte!).

Vorsicht beim Umbau

Bei allen Umbaumaßnahmen sollte man streng auf Sauberkeit achten. Besonders beim Einbau weiterer Schaltungen ist der PET 2001 vor dem Wiedereinschalten sorgsam nach Spänen, Lötklecksen u. a. abzusuchen – ein Staubsauger leistet hier gute Hilfe. Bereits eine eingedrungene Wellensittich-Feder hat – wohl zusammen mit kondensierter Luftfeuchtigkeit – für skurrile Daten gesorgt.

Zeichnen auf dem Bildschirm

Wer mit dem PET „malen" möchte, wird dies wohl vorzugsweise mit dem POKE-Befehl tun. Dazu benötigt er den Zeichencode für die grafischen Symbole des PET. Das Programm in *Bild 3* bringt sie auf den Schirm.

Michael Penzkofer

Binär-Dezimal-Umwandlung

Für die Umwandlung einer Zahl in ein anderes Zahlensystem, hier vom Binärsystem in das Dezimalsystem, gibt es prinzipiell zwei Lösungsmöglichkeiten: Entweder verwendet man die Zählmethode, wobei die in einer bestimmten Speicherzelle stehende Zahl binär abwärts bis auf Null und eine andere Speicherzelle von Null dezimal aufwärts gezählt wird. Letztere Zelle enthält dann das dezimale Ergebnis. Die zweite Möglichkeit ist die Verwendung eines geeigneten Algorithmus; dazu ist zwar erheblich weniger Rechenzeit, aber meist ein deutlich längeres Programm erforderlich.

Das im *Bild* aufgelistete Programm verwendet zwar auch einen bestimmten Umwandlungs-Algorithmus, ist aber trotzdem auffallend kurz. Eine besondere Eigenschaft ist, daß auch Zahlen verarbeitet werden, die größer als 99 (dezimal) sind.

Eine in der Zelle 00F5 stehende Binärzahl (00. . .FF) wird vom Programm in eine Dezimalzahl umgewandelt; das niederwertige Byte des Ergebnisses (LSB), erscheint an der Adresse 00F3, das höherwertige Byte (es kann 0, 1 oder 2 sein) an der Adresse 00F4 (MSB).

```
BINAER-DEZIMAL-UMWANDLUNG
MIT DEM 6502

0000 F8       SED
0001 A0 08    LDY #08
0003 A9 00    LDA #00
0005 85 F4    STA F4
0007 85 F3    STA F3
0009 06 F5    ASL F5
000B A5 F3    LDA F3
000D 65 F3    ADC F3
000F 85 F3    STA F3
0011 26 F4    ROL F4
0013 88       DEY
0014 D0 F3    BNE 0009
0016 D8       CLD
0017 60       RTS

ARGUMENT: 00F5
ERGEBNIS: 00F4,00F3
```

Zwar kommt man erst nach einiger Überlegung darauf, wie's funktioniert; die gewählte Konversionsmethode ist aber dafür recht schnell

Nach dem Durchlaufen des Programms sind Akku, Y-Register und der Inhalt der Zelle 00F5 verloren.

Zusammen mit dem in Heft 22/1978 vorgestellten Programm zur Analog/Digital-Umsetzung (KIM als Speicheroszilloskop) läßt sich aus dem Mikrocomputer KIM-1 ein kleines Digitalvoltmeter machen – aber das ist natürlich nicht die einzige Anwendungsmöglichkeit des hier beschriebenen Programms. *Fe.*

Stichwortsuche in BASIC

In FUNKSCHAU 1978, Heft 24, war unter dem Titel „KIM auf Datensuche" ein Stichwort-Suchprogramm beschrieben, das sich ideal für eine Software-Kartei für Adressen, Telefonnummern oder Literaturverzeichnisse eignet. Selbstverständlich läßt sich ein solches Programm auch in BASIC erstellen, wenn man die deutliche langsamere Suchgeschwindigkeit in Kauf nimmt. Die Daten, in denen nach dem einzugebenden Stichwort gesucht werden soll, sind als Strings in DATA-Blöcken (im *Bild* bei der Zeilennummer 500) zusammengefaßt. Wird ein Stichwort irgendwo innerhalb eines Strings gefunden, so wird der gesamte String ausgegeben. Bei einem Telefonnummern-Verzeichnis könnte der Data-Block etwa so aussehen:

500 DATA HUBER 1234567, MUELLER 7654321, MEIER 432123 usw.

Das Programm zeigt recht deutlich die Anwendung von MID $ (D $, I, K). I gibt dabei den Ort innerhalb des Strings an, wo gesucht wird, und K die Länge des Buchstaben-Vergleichs, die mit der Stichwort-Länge identisch ist. *Fe.*

Dieses BASIC-Programm – es wurde auf einem AIM-65 entwickelt – sucht in DATA-String-Blöcken nach einem einzugebenden Stichwort und eignet sich daher recht gut als Karteiprogramm

```
80 RESTORE:INPUT"STICHWORT";A$:K=LEN(A$)
90 READD$:IF D$="END" THEN 80
100 S=LEN(D$)-K:IF S<0 THEN 90
110 FOR I=1 TO S+1:M$=MID$(D$,I,K)
120 IF M$=A$ THEN 150
130 NEXT
140 GOTO90
150 PRINTD$:GOTO 90
500 DATA**MIT END ABSCHLIESSEN!**
READY.
```

Hans Werner

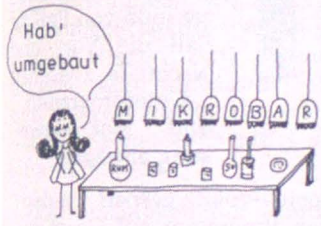
Mikrocomputer

Eine kleine Einführung

25. Teil

Zwei Modelle und ein Programm

Wir erinnern uns an das MIKROBAR-Programm. Es wurde mit den 8 Befehlen des ersten Modells geschrieben und bestand aus 14 Programmzeilen.



Nutzen wir die Befehle des zweiten Modells aus, verkürzt es sich auf 6 Zeilen. Das ist der Vorteil einer großen Befehlsmenge.

Die Tabelle verdeutlicht die Abfrage: Sind alle Stellen ausgegeben ?

```
→LA AC k    1D    Zahl 1 laden
→SP AC d   512D   Ausgabe : 1,2,4...
 SH LI d    1D    Schieben: 2,4,8...
 UN   k    255D   Abfrage : Sind alle
 BR EX d    ...   Stellen ausgegeben?
 JMP  d    ...    zum Anfang
```

Makros

Beide MIKROBAR-Programme haben einen Nachteil. Das Licht rotiert zu schnell. Abhilfe schafft eine zusätzliche

Warteschleife, so daß jede Lampe für die Zeit von vielen Programmdurchläufen leuchtet.

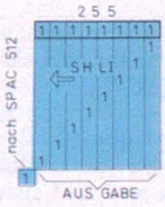
Das geht ohne Schwierigkeiten mit einem Unterprogramm. Viele Mikrocomputer bieten eine weitere Möglichkeit, die sogenannte Makroanweisung oder kurz MAKRO.

MAKROS sind Befehlsfolgen, denen per Definition ein Name zugewiesen wird.

Die Befehlsfolge:

```
LA AC d  515D
SP AC d  320D
AD    d  400D
SP AC d  512D
```

ist z. B. häufiger in einem Programm vorhanden. Ein besonders fauler Programmierer könnte doch auf die Idee kommen, die 4 Befehle einfach OTTO zu nennen und sie dann von seiner Sekretärin nachtragen zu lassen... oder nicht?



Glücklicherweise ist dies nicht notwendig. Das besorgt automatisch das Übersetzungsprogramm: ASSEMBLER in Maschinencode. Ohne weitere Assembleranweisungen geht das natürlich nicht.

Die erste Anweisung eines Makros ist der Begriff MAKRO. Danach folgt ein frei wählbarer Name, in unserem Fall OTTO. Die letzte Anweisung kennzeichnet das Ende (MEND = MAKRO Ende).

```
MAKRO
OTTO
LA AC d  512D
SP AC d  320D
AD    d  400D
SP AC d  512D
MEND
```

Die Begriffe MAKRO und MEND sind natürlich nicht genormt. Sie hängen vom Typ des Prozessors ab. Immer, wenn im Hauptprogramm OTTO steht, fügt das Übersetzungsprogramm alle 4 Befehle ein. Steht im Hauptprogramm dreimal OTTO, werden auch dreimal die 4 Befehle in das Maschinenprogramm eingesetzt.

Die Makroanweisung verringert lediglich die Schreibarbeit des Programmierers.

Sie verkürzt zwar das Quellenprogramm, nicht aber das Objektprogramm (Maschinencode)! Die Befehle MAKRO und MEND betreffen nur den Assembler und werden nicht in das Objektprogramm übernommen.

Vergleich: Unterprogramme und Makros

Jedes Unterprogramm ist nur einmal im Maschinenprogramm enthalten. Vom Hauptprogramm wird zum Unterprogramm gesprungen, dieses abgearbeitet und danach das Hauptprogramm an der alten Stelle fortgesetzt. Mehrfach

benutzte Unterprogramme verringern den Speicherbedarf, denn das Programm wird kürzer. Die Laufzeit des Programms erhöht sich geringfügig, weil die Rücksprungadresse umgespeichert werden muß. Das ist zusätzliche Arbeit.

Die Makroanweisung ist eigentlich nur eine verkürzte Schreibweise. Sind in einem Programm Makroanweisun-

gen vorhanden, ergibt ein Assemblerbefehl mehrere Befehle im Maschinencode. Bei Unterprogrammen ist das

nicht der Fall. Das Maschinenprogramm kann durch Makros nicht verkürzt werden, d. h. es kann kein Speicherplatz eingespart werden. Makroanweisungen werden schneller als gleichartige Unterprogramme abgearbeitet. Es entfällt das Retten der Rücksprungadresse.

Beispiel

Das MIKROBAR-Programm von Punkt B63 soll durch eine Zeitschleife vervollständigt werden. Den Ausgabebefehl SP AC 512 ersetzen wir durch die Makroanweisung ZEIT. Vor JMP wird ebenfalls das Makro ZEIT eingefügt, damit nach jedem vollständigen Programmdurchlauf eine Pause entsteht.

```
HAUPTPROGRAMM
    LA AC k    1D     Zahl 1 laden
    ZEIT              Makro ZEIT einfügen
    SH LI d    1D     Schieben: 2,4,8...
    UN    k    255D   Abfrage : Sind alle
    BR EX d    ...    Stellen ausgegeben?
    ZEIT              Makro ZEIT einfügen
    JPM   d    ...    zum Anfang

MAKRO
    MAKRO
    ZEIT              Makroanweisung ZEIT
                      definieren
    SP AC d    512D   Ausgabe: 1,2...bzw.0
    SP AC d    Z      1,2... für HP retten
    LA AC k    200D   200 Durchläufe bil-
    SP AC d    TIME   den die eigentliche
    IN    d    TIME   Zeitschleife
    BR EX d    ...
    LA AC d    Z      1,2... in AC laden
    MEND              Makro Ende
```

Erscheint bei der Übersetzung in den Maschinencode der Name ZEIT, werden jedesmal die 7 Befehle des Makros eingesetzt. Danach besteht das Programm aus 21 Befehlen. Hätten wir die Zeitschleife als Unterprogramm geschrieben, würde das Maschinenprogramm nur aus 16 Befehlen bestehen. RUF und POP sind im Gegensatz zu MAKRO und ZEIT keine Assembleranweisungen und bleiben erhalten.

Der Programmierer muß von Fall zu Fall die Vor- und Nachteile zwischen Unterprogramm und Makroanweisung abwägen:

1. Soll Speicherplatz gespart werden?
2. Soll das Programm auf Geschwindigkeit getrimmt werden?
3. Was macht weniger Arbeit?

(Fortsetzung folgt)

Fernschreiber als AIM-65-Drucker

Der im Mikrocomputer AIM-65 eingebaute Thermodrucker stellt lediglich 20 Zeichen pro Zeile als 5x7-Punktmatrix dar, was oft unbefriedigend ist, besonders, wenn man mit dem AIM-Editor Texte verarbeiten möchte.

Eine Alternative ist die Verwendung eines gewöhnlichen und oft preisgünstig erhältlichen Baudot-Postfernschreibers. Er kann über ein 5-V-/24-mA-Relais zwischen Pin A–U und + 5 V betrieben werden und läßt sich dann mit der F-3-Taste des AIM ein- und ausschalten. Für die Zeichenausgabe sorgt das Programm im *Bild*; die Einsprungadresse dieser Subroutine ist 0F39. Das bei 0F00 stehende Programm sorgt lediglich für die Initialisierung der AIM-Monitor-Speicherzellen und springt nach dem Start bei 0F00 zum Monitorprogramm zurück.

Das Baudot-Ausgabe-Unterprogramm bei 0F39 arbeitet auch mit Assembler- und BASIC-ROMs zusammen. Allerdings ist es dann sinnvoll, die verwendeten Zero-Page-Adressen höher zu legen, z. B. ab 00F0.

Hans-Joachim Regge

```
<M>=0F00 A9 1F 8D 13 | < > 0F40 01 0F 09 02 | < > 0F80 0F 20 91 0F | < > 0FC0 01 03 19 0E
< > 0F04 01 A9 4C 8D | < > 0F44 4C 51 0F C9 | < > 0F84 88 10 EE 48 | < > 0FC4 09 01 0D 1A
< > 0F08 17 A4 A9 00 | < > 0F48 5D F0 F7 29 | < > 0F88 00 A9 0F 20 | < > 0FC8 14 06 0B 0F
< > 0F0C 8D 18 A4 A9 | < > 0F4C 3F AA BD C0 | < > 0F8C 03 8C 0C 82 | < > 0FCC 12 1C 0C 18
< > 0F10 97 8D 06 A4 | < > 0F50 0F 85 AC 29 | < > 0F90 0F 08 00 0F | < > 0FD0 16 17 0A 05
< > 0F14 A9 0F 8D 07 | < > 0F54 20 C5 AB F0 | < > 0F94 EC 68 60 20 | < > 0FD4 10 07 1E 13
< > 0F18 84 20 FA E6 | < > 0F58 0E 85 AB A8 | < > 0F98 9E EB 85 A7 | < > 0FD8 1D 15 11 2B
< > 0F1C 0C 82 81 20 | < > 0F5C F0 04 A9 1B | < > 0F9C 29 7F 20 39 | < > 0FDC 00 3A 00 00
< > 0F20 F1 E6 20 1D | < > 0F60 D0 02 A9 1F | < > 0FA0 0F A5 A7 20 | < > 0FE0 24 2D 3A 31
< > 0F24 0F A9 05 8D | < > 0F64 20 6D 0F A5 | < > 0FA4 AC EB 4C 05 | < > 0FE4 14 19 34 25
< > 0F28 06 A4 A9 8F | < > 0F68 AC 20 6D 0F | < > 0FA8 EF 48 AD 00 | < > 0FE8 2F 32 2B 31
< > 0F2C 8D 07 A4 A9 | < > 0F6C 60 20 B5 0F | < > 0FAC 88 09 04 8D | < > 0FEC 2C 23 3C 3D
< > 0F30 00 8D 13 01 | < > 0F70 20 91 0F A0 | < > 0FB0 00 A8 68 60 | < > 0FF0 36 37 33 21
< > 0F34 4C 82 E1 00 | < > 0F74 04 4A 90 06 | < > 0FB4 00 48 AD 00 | < > 0FF4 2A 30 35 27
< > 0F38 00 C9 0D D0 | < > 0F78 20 A9 0F 4C | < > 0FB8 A8 29 FB 8D | < > 0FF8 26 38 2E 25
< > 0F3C 0A 89 08 00 | < > 0F7C 81 0F 20 B5 | < > 0FBC 00 A8 68 60 | < > 0FFC 2F 3E 32 39
```

Hex-Dump des Unterprogramms zur Baudot-Ausgabe (ab 0F39) und des Initialisierungs-Programms (ab 0F00)

Michael Konz

6502-Rechenroutinen

Will man einem Mikrocomputer das Rechnen beibringen, so stellen die vier Grundrechenarten bekanntlich die größten Probleme dar. Die folgenden Routinen befassen sich daher auch nur mit Addieren und Subtrahieren, Multiplizieren und Dividieren.

Das nachfolgend beschriebene Programm wurde für den AIM-65 geschrieben, der gewissermaßen als Nachfolger des KIM bezeichnet werden kann. Natürlich laufen die Routinen auch auf diesem, wenn geringfügige Änderungen, auf die später noch eingegangen werden wird, vorgenommen werden. Eine alphanumerische Ein- und Ausgabemöglichkeit wäre jedoch nützlich, wenn auch nicht unbedingt erforderlich.

Die Genauigkeit der Routinen läßt sich durchaus mit der guter Taschenrechner vergleichen. In den meisten Fällen liegt sie sogar höher. Es können Zahlen im Bereich von $1 \cdot 10^{-99}$ bis $9{,}999999999 \cdot 10^{99}$ verarbeitet werden. Intern wird mit einer zwölfstelligen Mantisse gearbeitet. Diese Stellen können bei entsprechender Parametereingabe auch alle angezeigt werden. Dennoch ist solch eine Programmierung nicht zu empfehlen, will man nicht auf Schutzstellen verzichten.

Bevor die einzelnen Routinen nun besprochen werden, soll eine Auflistung aller dem Anwender zur Verfügung stehenden Unterprogramme gegeben werden.

0200
Der Register-Pointer wird abhängig vom Inhalt der beiden Indexregister gesetzt.

0233
Ein Register wird in ein anderes kopiert. Angabe von Ursprungs- und Zielregister erfolgt durch die beiden Indexregister der CPU.

0240
Das x-Register wird in das y-Register kopiert.

0246
Das y-Register wird in das x-Register kopiert.

0259
Das x-Register wird mit dem y-Register vertauscht.

0269
Das durch das Indexregister X angegebene Register wird gelöscht.

027C
Exponent und Vorzeichen des durch den Index X angegebenen Registers werden gelöscht.

028D
Der Exponent im x-Register wird um 1 erhöht. Tritt ein Überlauf auf, so erfolgt eine Fehlermeldung und Rückkehr zum Monitor.

02AB
Der Exponent im x-Register wird um 1 erniedrigt; sonst wie 028D.

02BB
Die Mantisse im x-Register wird um 1 nach links verschoben, was einer Multiplikation mit 10 gleichkommt. Ein dabei auftretender Übertrag bleibt unberücksichtigt.

02C7
Die Zahl im x-Register wird in die Exponentialform gebracht.

02ED
Die Exponenten der Zahlen in x und y werden gleichgesetzt.

0354
Die Mantisse von x wird um 1 nach rechts verschoben, was einer Division durch zehn gleichkommt. Die letzte Ziffer geht verloren, der Exponent bleibt unverändert.

0369
Wie 0354, jedoch für das y-Register.

037E
Der Inhalt des x-Registers wird ausgegeben.

0437
Die Inhalte des x- und des y-Registers werden verglichen, das Ergebnis des Vergleiches an der Zero-Page-Adresse F2 gespeichert.

044C
Wie 0437, es werden aber nur die absoluten Beträge verglichen.

0464
Wie 0437, es werden jedoch nur die Mantissen verglichen.

04B8
Die Inhalte von x und y werden addiert, das Ergebnis im x-Register gespeichert.

0518
Die Inhalte von x und y werden subtrahiert (x–y), das Ergebnis im x-Register gespeichert.

0523
Eingabe einer Zahl in das x-Register.

05CF
Die Inhalte von x und y werden multipliziert, das Ergebnis im x-Register gespeichert.

069A
Der Inhalt von x wird durch den Inhalt von y dividiert, das Ergebnis in x gespeichert.

0766
Ausgabe des Textes „UNVERSTAENDLICHE EINGABE".

0768
Ausgabe eines beliebigen alphanumerischen Textes.

077A
Ausgabe des Textes „RECHENKAPAZITAET UEBERSCHRITTEN" und Rückkehr zum Monitor.

Um Verwirrungen zu vermeiden, muß zunächst einmal auf die Register eingegangen werden. Die von den Routinen benutzten Register dienen zur Aufnahme der Daten und belelgen je 16 Byte im Speicher. Sie dürfen nicht mit den beiden Indexregistern der CPU verwechselt werden.

In der Zero-Page stehen insgesamt 15 Register zur Verfügung. Davon werden neun von den Routinen belegt, die übrigen kann der Programmierer zur eigenen Datenspeicherung verwenden. Die beiden ersten Register erhielten den Namen x und y. Sie sind an den Adressen 0000-000F und 0010-001F zu finden. Weiterhin kann der Anwender ab einer von ihm bestimmbaren Page beliebig viele zusätzliche Register festlegen. Dies geschieht durch Schreiben der Zero-Page-Adresse F4. Der Wert 0A bezeichnet dann den Speicherraum ab der Adresse 0A00. Dieser Parameter ist vor dem ersten Aufruf einer Transfer-Routine einzugeben. Natürlich muß dabei darauf geachtet werden, daß die Daten nicht das Programm zerstören,

```
0200 8A 20 18 02 85 FB 20 2D 02 85 FA 98 AA 20 18 02
0210 85 FD 20 2D 02 85 FC 60 29 F0 C9 F0 D0 04 A9 00
0220 F0 0A 8A 4A 4A 4A 4A D8 18 65 F4 F8 60 8A 0A 0A
0230 0A 0A 60 20 00 02 A0 0F B1 FA 91 FC 88 10 F9 60
0240 A2 F0 A0 F1 30 ED A2 F1 A0 F0 30 E7 A2 F0 A0 F2
0250 20 33 02 A2 F1 A0 F3 30 DA A2 F1 A0 FD 20 33 02
0260 20 40 02 A2 FD A0 F0 30 CA 20 00 02 A0 0F A9 00
0270 91 FA 88 10 FB A9 01 A0 0D 91 FA 60 20 00 02 A0
0280 0C A9 00 91 FA C8 C8 91 FA C8 91 FA 60 F8 A5 0E
0290 30 0D A5 0F 18 69 01 85 0F 90 03 4C 7A 07 60 A5
02A0 0F 38 E9 01 85 0F D0 F6 85 0E 60 F8 A5 0F D0 05
02B0 E6 0F C6 0E 60 24 0E 30 DB 10 E6 A2 00 B5 01 95
02C0 00 E8 E0 0B 90 F7 60 F8 A0 00 A5 00 F0 0F A5 0D
02D0 C9 01 D0 01 60 C6 0D 20 8D 02 B8 50 F1 20 BB 02
02E0 20 AB 02 C8 C0 0C 90 E2 A2 F0 4C 7C 02 A5 0E C5
02F0 1E F0 1F 24 0E 10 0D A5 0F F0 44 20 54 03 20 8D
0300 02 B8 50 F3 A5 1F D0 04 E6 1E 10 2A 20 62 03 B8
0310 50 F2 A5 0F C5 1F D0 01 60 24 0E 10 17 90 0C 20
0320 4B 03 20 54 03 20 8D 02 B8 50 F4 20 4B 03 20 62
0330 03 B8 50 F7 90 09 20 4B 03 20 77 03 B8 50 F7 20
0340 4B 03 20 54 03 20 8D 02 B8 50 F4 A5 0F C5 1F D0
0350 02 68 68 60 A2 0A B5 00 95 01 CA 10 F9 A9 00 85
0360 00 60 A5 1F 38 E9 01 85 1F A2 0A B5 10 95 11 CA
0370 10 F9 A9 00 85 10 60 A5 1F 18 69 01 90 E9 20 44
0380 EB A9 03 85 F0 A9 00 85 F1 A2 F0 A0 FD 20 33 02
0390 A5 F1 30 23 A5 0F C9 10 B0 1D 24 0E 30 0C A5 0F
03A0 F0 15 E6 0D 20 AB 02 B8 50 F4 A5 0F F0 09 20 54
03B0 03 20 8D 02 B8 50 F3 D8 18 A5 0D 65 F0 F8 C9 0C
03C0 90 02 A9 0B AA 86 F0 B5 00 C9 05 90 24 CA EA 18
03D0 B5 00 69 01 48 0A 0A 0A 0A 68 29 0F 95 00 90 11
03E0 CA 10 EC E6 0D 20 54 03 E6 00 A5 F1 10 03 20 C7
03F0 02 A2 00 A5 0C 10 04 A9 2D 10 02 A9 20 20 7A E9
0400 B5 00 20 51 EA E8 E4 F0 B0 08 E4 0D D0 F2 A9 2E
0410 10 EB A5 0F F0 1A 20 3E E8 A9 45 20 7A E9 A5 0E
0420 10 04 A9 2D 10 02 A9 2B 20 7A E9 A5 0F 20 46 EA
0430 A2 FD A0 F0 4C 33 02 A5 0C C5 1C 90 07 F0 09 A9
0440 80 85 F2 60 A9 40 10 F9 A5 0C 30 37 A5 0E C5 1E
0450 90 F2 F0 02 B0 E9 A5 0E 30 1F A5 0F 30 1B 90 DF
0460 F0 02 B0 E0 A2 00 B5 00 D5 10 90 D3 F0 02 B0 D4
0470 E8 E0 0C 90 F1 A9 00 F0 C8 A5 0F C5 1F 90 C5 F0
0480 E3 B0 BC A5 0E C5 1E 90 B6 F0 02 B0 B7 A5 0E 30
0490 0A A5 0F C5 1F 90 AD F0 0C B0 A4 A5 0F C5 1F 90
04A0 9E F0 02 B0 9F A2 00 B5 00 D5 10 90 97 F0 02 B0
04B0 8E E8 E0 0C 90 F1 B0 BD 20 ED 02 A5 0C C5 1C D0
04C0 21 A2 0B 18 B5 00 75 10 48 0A 0A 0A 0A 68 29 0F
04D0 95 00 CA 10 EF 90 07 20 54 03 E6 0D E6 00 20 C7
04E0 02 60 20 64 04 A5 0C 10 08 24 F2 70 1F E6 0C 10
04F0 06 24 F2 70 17 C6 0C 38 A2 0B B5 10 F5 00 29 0F
0500 95 00 CA 10 F5 B0 D7 20 54 03 F0 D2 38 A2 0B 20
0510 B4 05 EA CA 10 F9 30 ED 20 ED 02 A5 0C C5 1C D0
0520 A0 F0 BF A2 0F A9 00 95 00 CA 10 FB 20 F0 E9 4E
0530 11 A4 A0 00 20 5F E9 C8 C9 0D D0 F8 AE 15 A4 9D
0540 38 A4 0E 11 A4 A2 00 A0 00 B9 38 A4 C9 0D F0 2F
0550 C9 2D D0 05 C6 0C C8 10 F0 C9 2E D0 08 86 0D A5
0560 0D F0 17 D0 F1 C9 45 F0 23 20 7D EA B0 0C C9 0A
0570 B0 08 95 00 E8 C8 E0 0B 90 CF 20 66 07 50 A4 A5
0580 0D D0 02 86 0D 20 C7 02 4C 7E 03 EA C8 B9 38 A4
0590 C9 2D D0 06 C6 0E C8 B9 38 A4 C9 3A B0 DC 20 84
05A0 EA B0 D7 C8 B9 38 A4 C9 3A B0 CF 20 84 EA B0 CA
05B0 85 0F 90 CB B5 00 F5 10 29 0F 95 00 60 A5 FF C6
05C0 FE 30 09 C6 FE 30 07 18 65 FF 90 F7 A9 00 60 20
05D0 4C 02 A9 00 85 FA 20 6C 02 A9 10 85 FA 20 6C 02
05E0 A2 00 D8 A9 0B 86 F8 38 E5 F8 A8 F8 B5 30 F0 50
05F0 85 FF A2 0B B9 20 00 85 FE 20 BD 05 48 29 0F 18
0600 75 10 48 0A 0A 0A 0A 68 29 0F 95 10 68 08 CA 10
0610 12 48 20 69 03 68 4A 4A 4A 4A 28 65 10 85 10 E6
0620 1D 10 0C 4A 4A 4A 4A 28 75 10 95 10 88 10 C5 20
0630 59 02 20 C7 02 20 59 02 20 B8 04 A2 F1 20 69 02
0640 A6 F8 E8 E0 0C 90 9B 20 4D 06 4C C7 02 A5 0F F0
0650 04 C6 0F E6 0D A5 2C C5 3C F0 02 C6 0C A5 2E C5
0660 3E D0 0F 85 0E 18 A5 2F 65 3F 90 03 4C 7A 07 85
0670 0F 60 A5 2F C5 3F 24 2E 10 12 90 08 38 E5 3F 85
0680 0F C6 0E 60 A5 3F 38 E5 2F 85 0F 60 90 06 20 7C
0690 06 E6 0E 60 20 84 06 C6 0E 60 A2 F0 A0 F4 20 33
06A0 02 A2 F1 A0 F5 20 33 02 A2 F1 20 7C 02 A2 F0 20
06B0 69 02 20 37 04 A5 F2 D0 0B A2 FF 9A A2 BE 20 68
06C0 07 4C 82 E1 A2 F4 A0 F0 20 33 02 A2 F0 20 7C 02
06D0 A2 F0 A0 F7 20 33 02 A2 F6 20 69 02 A2 00 86 F7
06E0 86 F9 20 18 05 A5 0C 30 04 E6 F7 10 F5 A5 F7 A6
06F0 F9 95 60 E8 E0 0C B0 3F 86 F9 48 20 46 02 20 C7
0700 02 A2 F1 20 69 02 68 85 10 20 CF 05 20 40 02 A2
0710 F7 A0 F0 20 33 02 20 18 05 A2 F5 A0 F1 20 33 02
0720 A2 F1 20 7C 02 E6 0D 20 C7 02 A2 F0 A0 F7 20 33
0730 02 A2 00 86 F7 F0 AB A2 F6 A0 F0 20 33 02 20 44
0740 07 4C C7 02 A2 F4 A0 F2 20 33 02 A2 F5 A0 F3 20
0750 33 02 A5 2C C5 3C F0 02 C6 0C A5 2E C5 3E D0 03
0760 4C 72 06 4C 63 06 A2 85 20 F0 E9 BD 00 07 C9 02
0770 F0 06 20 7A E9 E8 D0 F3 B8 60 A2 9E 20 68 07 A2
0780 FF 9A 4C 82 E1 55 4E 56 45 52 53 54 41 45 4E 44
0790 4C 49 43 48 45 20 45 49 4E 47 41 42 45 02 52 45
07A0 43 48 45 4E 4B 41 50 41 5A 49 54 41 45 54 20 55
07B0 45 42 45 52 53 43 48 52 49 54 54 45 4E 02 52 45
07C0 53 55 4C 54 41 54 20 55 4E 45 4E 44 4C 49 43 48
07D0 02 03
```

Bild 1. Dieses hexadezimale Listing enthält alle AIM-65-Rechenroutinen

das sich an den Speicherplätzen 0200-07D0 befindet.

Innerhalb der Register sind die Daten folgendermaßen angeordnet:
XXX0-XXXB: 12stellige Mantisse

XXXC: Vorzeichen der Mantisse (00: positiv, FF: negativ)
XXXD: Stellung des Dezimalpunktes
XXXE: Vorzeichen des Exponenten
XXXF: Exponent

Nun zu den einzelnen Routinen. Zum Transfer eines Registers in ein anderes wird die Routine ab der Adresse 0233 verwendet. Das Ursprungsregister wird durch eine hexadezimale Ziffer angegeben, die vor dem Aufruf der Routine im Indexregister X gespeichert werden muß. Das Zielregister wird genauso im Index Y gespeichert. Dabei bezeichnen die Zahlen FX Register in der Zero-Page. F0 ist das x-Register, F1 das y-Register. Alle anderen Register beziehen sich auf die Angabe in F4. Ist dieser Parameter z. B. 0A, dann bezeichnet das Register 00 die 16 Byte an den Adressen 0A00...0A0F; 01 ist bei 0A10...0A1F zu finden usw.

Zu den Registern wäre noch zu sagen, daß das Register FF, das in der Zero-Page an den Adressen FO...FF stehen würde, nicht benutzt werden kann, da die FX-Adressen für den Programmablauf wichtige Parameter enthalten.

Für die Routinen 0269 und 027C gilt das oben Gesagte für die Adressierung der Register. Es wird jedoch nur ein Wert im Index X benötigt.

Die Routine 0200 schließlich dient dazu, den Pointer auf die angegeben Register zu setzten. Dieser Pointer befindet sich an den Adressen FA und FB bzw. FC und FD. Normalerweise wird diese Routine nicht benötigt, da eine Transferroutine ihren Aufruf beinhaltet.

Die Routine 02C7 gehört zu den wichtigsten überhaupt. Sie sorgt dafür, daß alle Daten nach jeder Eingabe und nach jeder Rechenoperation immer in Exponentialform vorliegen:
$1{,}23456789012 \cdot 10^{99}$

Die Routine 02ED ist nicht weniger wichtig. Sie gleicht die Exponenten im x- und y-Register aneinander an. Dazu ein Beispiel:

Im x-Register steht die Zahl $1{,}23 \cdot 10^{08}$,
im y-Register die Zahl $5{,}67 \cdot 10^{05}$.

Nach Aufruf der Routine steht die Zahl im x-Register unverändert, im y-Register steht jedoch $0{,}00567 \cdot 10^{08}$. Es ist sofort zu sehen, daß der Wert derselbe ist wie zuvor.

Zur Ausgabe von Daten stehen zwei Formate zur Verfügung. Wird die Routine 037E aufgerufen, so erfolgt die Anzeige mit Festkomma, das auf drei Stellen gesetzt ist, soll die Anzeige im Ex-

```
0800 A2 FC A0 F0 20 33 02 A2 F9 A0 F1 20 33 02 20 CF
0810 05 A2 F0 A0 FC 20 33 02 A2 F0 20 69 02 A2 FA A0
0820 F1 4C 33 02 20 C7 02 20 59 02 20 CF 05 A2 F0 A0
0830 FA 20 33 02 A9 00 85 AC 20 40 02 A2 FC A0 F0 20
0840 33 02 20 9A 06 20 40 02 A2 F8 A0 F0 20 33 02 20
0850 B8 04 A2 F0 A0 F8 20 33 02 4C 00 08 20 23 05 A2
0860 F0 A0 F8 20 33 02 20 40 02 20 CF 05 A2 F0 A0 F9
0870 20 33 02 A2 FA 20 69 02 E6 A0 A2 F8 A0 FC 20 33
0880 02 A9 00 85 F6 20 00 08 20 99 08 C6 0C 20 24 08
0890 20 99 08 20 24 08 4C 88 08 A6 F6 E0 15 90 0F 68
08A0 68 A2 F8 A0 F0 20 33 02 20 F0 E9 4C 7E 03 A0 00
08B0 BD C3 08 99 00 00 E8 C8 C0 03 90 F4 A9 02 85 0F
08C0 86 F6 60 00 00 06 00 02 00 00 04 02 00 07 02 01
08D0 01 00 01 05 06 02 01 00 20 5C 08 20 5F E9 4C D8
08E0 08
```

Bild 2. Berechnung des Sinus, Startadresse ist 08D8

ponentialformat erfolgen, so müssen zuvor die Zero-Page-Adressen F0 und F1 beschrieben werden. Normalerweise stehen hier die Werte 03 und 00. Für das Exponentialformat muß der Wert von F1 zu FF geändert werden. An F0 muß dann die Zahl der Nachkommastellen geschrieben werden. Die Startadresse lautet dann 0389. In beiden Fällen findet eine automatische 5/4-Rundung statt. Vorsicht: Beim Aufruf der Routine wird das Register FD überschrieben.

Die nächsten drei Routinen sind eigentlich eine einzige mit drei verschiedenen Startadressen. Wie bereits gesagt, dienen sie zum Vergleich des x- und des y-Registers. Das Resultat des Vergleiches hat folgende Form:

Ist x kleiner als y, dann wird Bit 7 von F2 gesetzt.

Ist x größer als y, dann wird Bit 6 von F2 gesetzt.

Ist x gleich y, dann wird F2 mit dem Wert 00 geschrieben.

Das Ergebnis des Vergleiches kann also leicht mit der Instruktion BIT F2 abgefragt werden, gefolgt von BMI, BPL, BVC oder BVS, beziehungsweise von der Befehlsfolge LDA F2; BEQ/BNE.

Die Routine 04B8 ist die erste wirkliche Rechenroutine. 0523 dient zur Übernahme von Daten ins x-Register. Bei der Eingabe ist folgendes zu beachten: Das Vorzeichen ist immer vor der Zahl einzugeben (Im Gegensatz zu den meisten Taschenrechnern). Die Exponenteneingabe wird mit einem E begonnen. Der Exponent muß immer zweistellig eingegeben werden. Die Eingabe –123E-8 würde zu einer Fehlermeldung führen. Richtig muß es heißen: –123E-08.

Eine fehlerhafte Eingabe kann durch Betätigen der DEL-Taste rückgängig gemacht werden, solange nicht RETURN gedrückt wurde. Die zulässige Stellenanzahl kann durch den Wert an der Adresse 0577 bestimmt werden. Eine Stellenzahl von 10 ist vorgesehen. Mehr Ziffern führen nach RETURN zu einer Fehlermeldung. Zu beachten ist, daß nach einer Fehlermeldung die Eingabe automatisch wiederholt wird, bis eine korrekte Eingabe erfolgt.

Sehr nützlich ist die Routine 0768, die dem Anwender erlaubt, eigene Erläuterungen zu Rechenoperationen ausgeben zu lassen. Es stehen hier noch 47 Byte für beliebige ASCII-Zeichen zur Verfügung. Diese werden ab der Adresse 07D1 bis maximal 07FF gespeichert. Jeder Text wird mit CNTRL B (Code 02) beendet. Vor jedem Aufruf muß die Startadresse für den Text im Indexregister X gespeichert werden, d. h. das niederwertige Adreßbyte. Werden die drei Fehlermeldungen der Routinen vom Anwender abgekürzt, so steht ein größerer Raum für eigene Texte zur Verfügung. Dazu die Adressen:

06BD: Fehlermeldung bei Division durch 0

0767: Fehlermeldung bei fehlerhafter Eingabe

077B: Fehlermeldung bei Überschreiten der Rechenkapazität

Der Textbuffer beginnt bei der Adresse 0785.

Soviel zu den Routinen. Ein kleines Beispiel soll zeigen, wie aus ihnen komplexe Funktionen zusammengesetzt werden können. *Bild 2* zeigt ein Programm zur Berechnung des Sinus. Die Eingabe erfolgt im Bogenmaß, Startadresse ist 08D8.

Wie schon gesagt, sind einige Änderungen erforderlich, soll das Programm auf einem anderen System laufen. Die größten Schwierigkeiten dürften dabei bei der Eingabe-Routine auftreten, da der Display-Buffer des AIM als Zwischenspeicher benutzt wird. Nur so ist die Benutzung der DELETE-Funktion für Korrekturen möglich. Der Anwender muß hier ggf. nach einer gleichwertigen Lösung suchen oder auf die Korrekturmöglichkeit verzichten.

Ferner finden folgende Routinen des AIM-Monitors Verwendung:

E97A

Ausgabe eines ASCII-Zeichens im Akku.

EA51

Ausgabe der vier niederwertigen Bits des Akku als ASCII-Zeichen.

EA46

Wie EA51, jedoch werden zuvor die vier höherwertigen Bits ausgegeben.

EA7D

Ein ASCII-Zeichen im Akku wird in eine hexadezimale Ziffer von 0-F verwandelt.

EA84

Wie EA7D, jedoch werden bei zweimaligem Aufruf zwei ASCII-Zeichen in ein Byte gepackt. In beiden Fällen wird das Carry-Flag gesetzt, wenn es sich bei dem Zeichen um keine hexadezimale Ziffer handelt.

E9F0

Ausgabe einer CRLF-Folge bzw. Löschen des Display.

E95F

Eingabe eines ASCII-Zeichens von der Tastatur in den Akku.

Abschließend noch einige Worte zur Rechengeschwindigkeit. Hier die Werte für die verschiedenen Operationen:

Addition: 0,5...3,5 ms
Subtraktion: 0,5...3,5 ms
Multiplikation: 4,5...60 ms
Division: 140...180 ms

Die Werte gelten für eine Taktfrequenz von 1 MHz. Diese Geschwindigkeiten erscheinen recht langsam, vergleicht man sie mit denen guter programmierbarer Taschenrechner. Die Überlegenheit des Mikrocomputers zeigt sich erst in langen Programmen, da zwischen der Ausführung zweier Operationen nur wenig Mikrosekunden vergehen. Dadurch wird die relativ geringe Rechengeschwindigkeit mehr als ausgeglichen.

Als letztes sollte noch gesagt werden, daß, wenn der Anwender innerhalb seines eigenen Programmes den Dezimalmodus mit dem Befehl CLD aufhebt, vor Aufruf der nächsten Rechenroutine unbedingt der gegenteilige Befehl, nämlich SED, folgen muß, sollen die Routinen einwandfrei arbeiten.

Erich H. Franke

KIM als Morse-Trainer

Ein drahtloser Plausch mit Australien, technische Gespräche mit Gleichgesinnten in den USA – diese und andere Wünsche wecken in zahlreichen Funkbegeisterten den Gedanken an eine eigene Kurzwellen-Sendestation. Ein Anwärter auf die begehrte Lizenzurkunde muß bei der Bundespost Kenntnisse über Technik und Gesetze nachweisen – und er muß morsen können.

Nun, alles läßt sich erlernen. Der Kandidat findet z. B. beim DARC, einer deutschen Amateurfunkvereinigung, Hilfe, unter anderem in Form eines Morsekurses auf Kassetten. Dieser enthält Morsetexte, die mitgeschrieben und mit der „Auflösung" im Begleitheft verglichen werden können. Die Texte bestehen teils aus Klarschrift, teils aber auch aus Gruppen zu je fünf Zeichen, die zufällig ausgewählt sind, um reines Auswendiglernen zu erschweren. Immer wieder kommt es jedoch vor, daß an Schulungsabenden in den Ortsverbänden die Teilnehmer klagen, sie hätten nach wie vor Probleme beim Mitschreiben der Zeichen, könnten die Übungstexte aber längst auswendig.

Das vorliegende Programm für den KIM-1 soll hier Abhilfe schaffen. Es erzeugt zufällige Fünfergruppen als Nf-Signal und protokolliert diese auf einem angeschlossenen Fernschreiber. Dabei können Zeichenmenge, Ausgabegeschwindigkeit und Zeichenabstand den Kenntnissen des Lernenden angepaßt werden. Es stehen ihm dazu elf Lektionen zur Verfügung. In der ersten werden nur vier, in der letzten alle 44 für die Prüfung notwendigen Zeichen in die Gruppen aufgenommen. Hilfen wie doppelter Wortabstand und stufenweise Steigerung der Geschwindigkeit sind ebenfalls vorhanden.

Das Programm läuft vollständig im KIM-1-System. Speichererweiterungen und Umbauten an der Platine sind nicht nötig. Da sowohl die Töne als auch die Fernschreibzeichen vom Prozessor erzeugt werden, genügt als Ausgabegerät die Schaltung nach *Bild 1*. Der Port PA 0 (Anschluß A-14 auf dem Applikationsstecker) dient als Ausgang für die Morsezeichen. Sie werden einem Kassettenrecorder zwecks Aufzeichnung zugeführt und gleichzeitig über den Lautsprecher hörbar gemacht. Das Fernschreib-Interface wird von PB 0

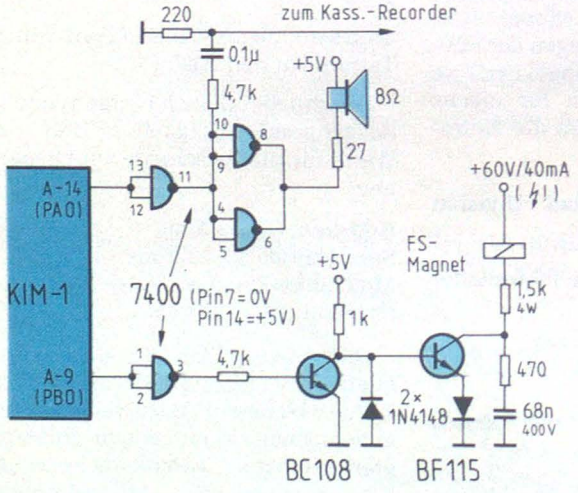


Bild 1. Einfache Interface-Schaltung für Lautsprecher, Kassettenrecorder und Fernschreiber

```
0000 40 80 03 05 2C 21 06 00 1B 02 FF 30 0E 05 42 42
0010 41 CA 2B 21 28 0D 45 4C 30 56 51 53 41 3D 54 32
0020 4F 43 44 35 52 2F 49 39 47 58 46 34 3F 54 4E 4A
0030 55 37 48 2C 4B 42 33 50 4D 59 5A 57 36 31 38 2E
0040 28 29 40 80 03 05 2C 21 06 22 02 00 00 36 4A 16
0050 00 00 00 7A B4 B6 44 14 CE 86 56 94 FC 7C 3C 1C
0060 0C 04 84 C4 E4 F4 E2 AA AC 8C 54 32 00 60 88 A8
0070 90 40 28 D0 08 20 78 B0 48 E0 A0 F0 68 D8 50 10
0080 C0 30 18 70 98 B8 C8 00 00 00 00 00 11 94 DC 91

0200 A2 06 B5 42 95 00 CA 10 F9 A2 01 8E 03 17 8E 01
0210 17 8E 02 17 CA 8E 00 17 A5 4A 48 A5 49 48 08 4C
0220 00 1C 20 DA 02 20 DA 02 A9 3C 20 73 02 A6 03 20
0230 92 02 A2 0F 86 0C A2 05 86 0D 38 A5 0F 65 12 65
0240 13 85 0E A2 04 B5 0E 95 0F CA 10 F9 29 3F C5 04
0250 B0 E8 AA B5 16 20 E1 02 20 73 02 C6 0D D0 DB A6
0260 03 20 92 02 20 DA 02 C6 0C D0 CB A9 3E 20 73 02
0270 4C 18 02 A8 B9 2A 00 38 F0 01 18 A0 02 A2 01 2A
0280 90 02 A2 03 20 94 02 A2 01 20 92 02 C9 80 D0 EA
0290 A6 02 A0 00 48 98 A4 01 84 14 8D 00 17 A4 00 8C
02A0 05 17 AC 05 17 10 FB C9 00 F0 02 49 01 C6 14 D0
02B0 E9 CA D0 E2 8E 00 17 68 60 A5 06 85 0A A5 05 38
02C0 E9 01 B0 02 C6 0A A4 0A 10 F5 60 A5 06 85 0A A5
02D0 05 4A 46 0A 90 E9 09 80 B0 E6 A9 0D 20 E1 02 A9
02E0 0A 85 0B 20 1F 03 85 07 86 09 20 B9 02 AD 02 17
02F0 29 FE 8D 02 17 20 B9 02 A2 05 AD 02 17 29 FE 46

0300 07 69 00 8D 02 17 20 B9 02 CA D0 EE AD 02 17 09
0310 01 8D 02 17 20 B9 02 20 CB 02 A6 09 A5 0B 60 86
0320 15 AA BD 5F 03 29 C0 F0 23 10 0E 0A 10 1B 20 54
0330 03 A9 0D 20 E6 02 4C 4C 03 24 08 30 0F A9 80 85
0340 08 A9 1F 20 E6 02 4C 4C 03 20 54 03 BD 5F 03 29
0350 1F A6 15 60 24 08 10 FB A9 1B 85 08 4C E6 02 00
0360 08 00 00 00 89 00 0B 00 04 02 00 00 08 1F 1B 00
0370 00 00 00 00 1F 00 00 00 00 00 09 00 00 00 00 04
0380 9C 85 9B 5F 89 91 85 8F 92 91 91 8C 83 9C 9D 96
0390 97 93 81 8A 90 95 87 86 98 8E 8E 8F 9E 92 99 5F
03A0 43 59 4E 49 41 4D 5A 54 46 4B 4F 52 5C 4C 58 56
03B0 57 4A 45 50 47 5E 53 5D 55 51 8F 9D 92 8B 83 FE
```

Bild 2. ▶
Hexadezimale Auflistung des Morseübungs-Programms

Tabelle 1: Wichtige Variablen des Morse-Trainers

0000	Tonhöhe der Morsezeichen
0001	Zeichenlänge
0002	Zeichenabstand
0003	Wortabstand
0004	Nummer der Lektion (00...2C)
0005	BAUDL
0006	BAUDH

Tabelle 2: Einstellen der Fernschreiber-Geschwindigkeit

Baud	BAUDH	BAUDL
45,45	06	21
50	05	80
75	03	C7
100	02	C9

(Anschluß A-9) gesteuert. Eine alternative Schaltung findet sich in FUNKSCHAU 1979, Heft 1.

Das Programm selbst (*Bild 2*) besteht aus zwei Blöcken. Es belegt mit allen für den Betrieb notwendigen Tabellen die Speicherzellen 0016 bis 008B sowie 0200 bis 03BE. Die Startadresse ist 0200.

Die Bedienung ist weitgehend unkritisch. Nach dem Start erscheint auf der Anzeige „0222 20", die Adresse des „Warmstart-Punktes". In der Zero Page sind sieben Speicherplätze für Programmparameter reserviert, die Zeichenlänge usw. bestimmen. Diese kann man nun nach Belieben abändern, um das Programm seinen Bedürfnissen anzupassen. *Tabelle 1* zeigt die Adressen dieser Variablen und ihre Wirkungen. Der Wert in BAUDH und BAUDL bestimmt die Schrittgeschwindigkeit des abgegebenen Fernschreibsignales. Hierfür ist ein Standardwert von 45,45 Baud eingesetzt; dieser muß gegebenenfalls für vorhandene Geräte abgeändert werden. *Tabelle 2* gibt darüber Auskunft.

Ist nun alles nach Wunsch geändert, so kommt man durch Druck auf PC wieder zur Anzeige „0222 20" und kann das Programm nun endgültig mit „GO" in Gang setzen. Auf dem Fernschreiber wird nun zweimal Wagenrücklauf-Zeilenvorschub ausgegeben, als Morsezeichen erscheint „Verkehrsbeginn" (ct). Daraufhin folgen 15 Fünfergruppen, die zufällig aus dem Lektionsstoff ausgewählt werden und mit dem „Schlußzeichen" (+) enden. Das Programm hält nun wieder bei „0222 20" an und kann – mit oder ohne neuerliche Parameter-Änderungen – beliebig oft mit „GO" wieder gestartet werden.

Das alles funktioniert so: Nach der Initialisierung bei 0200 wird mit dem Zufallsgenerator eine Zahl zwischen 1 und einem von der Lektion abhängigen Maximalwert erzeugt. Mit diesem Zeiger wird aus der Tabelle bei 0016 ein ASCII-Zeichen geholt, das – in Baudot gewandelt – zum Fernschreiber geschickt und als Morsezeichen auf Kassette gespeichert wird.

Das Fernschreiber-Ausgabeprogramm bei 02E1 ist ein Modul, das häufig benötigt wird. Es kann über die Tabelle bei 035F auch Symbole ausgeben, die im Fernschreib-Code selbst nicht enthalten sind. Diese werden als Folge zweier Symbole dargestellt. Das Programm bei 02DA gibt einen Wagenrücklauf/Zeilenvorschub aus. Stolze Besitzer einer Teletype oder eines anderen Ausgabegerätes können an dieser Stelle auch Sprünge zu eigenen Ausgabegeräten einsetzen.

Dieser Trainer ist vor allem auch für Ortsverbände interessant, die selbst Morseübungen abhalten und diese sinnvoll ergänzen wollen, zumal auf dieser Basis die Beschaffung der notwendigen Geräte heutzutage kaum mehr ein Problem darstellt.

Basic in 2 KByte

Üblicherweise, so könnte man meinen, belegt ein Basic-Interpreter etwa 8 KByte. Nimmt man jedoch einige kleine und in bestimmten Fällen unbedeutende Einschränkungen hin, so kommt man mit 2 KByte aus – nämlich mit dem „Tiny Basic", das von Tom Pittman für den KIM-1 und andere 6502-Systeme entwickelt wurde.

Tiny Basic existiert für den KIM-1 in zwei Versionen auf Kassette, nämlich für den Adressenbereich 2000...28E5 und für den Bereich ab 0200. Beide sind nun auch in Deutschland zu haben, nämlich von der Fa. Hofacker (Holzkirchen). Geht man von der erstgenannten Version aus, so läßt sich bei 2003 auch ein „Warmstart" durchführen, der den Programmspeicher nicht löscht. An der Adresse 200F kann das Terminal-ASCII-Zeichen für Back Space programmiert werden, üblicherweise hex 08. Aus einem laufenden Basic-Programm kann man z. B. durch Drücken der Space-Taste am Terminal „aussteigen", da Tiny Basic zyklisch den TTY-Eingang des KIM-1 abfragt.

An Systembefehlen stehen CLEAR (statt NEW), RUN, GOTO (wie RUN, jedoch Start an einer bestimmten Zeilennummer), und LIST in mehreren Varianten zur Verfügung. Basic-Befehle können, wenn man zu Beginn der Zeile eine Zeilennummer wegläßt, auch direkt ausgeführt werden.

Tiny Basic besitzt, wie erwähnt, einige Einschränkungen hinsichtlich des Befehlsvorrats. Folgende Befehle sind im Programm möglich: GOTO, GOSUB, RETURN, LET (kann entfallen), PRINT (Variable, arithm. Ausdruck oder Text), INPUT (Variable), IF...THEN (THEN kann entfallen) und END. Tiny Basic beherrscht die vier Grundrechenarten (+, –, *, /) und die Vergleiche =, >=, <=, >, <, <>, die zusammen mit IF verwendet werden. Die Rechenhierarchie ist „Punkt vor Strich". Die Kurzform von PRINT ist PR.

Gegenüber komfortablen Basic-Interpretern müssen folgende Einschränkungen hingenommen werden: Pro

```
:LIST
10 PR"WIEVIELE SEK.";
20 INPUT A
30 B=30*A
40 C=0
50 C=C+1
60 IF C<B GOTO 50
70 PR A;" SEK. ";
80 IF A=1 GOTO 110
90 PR"SIND UM"
100 GOTO 10
110 PR"IST UM"
120 GOTO 10

:RUN
WIEVIELE SEK.? 2
2 SEK. SIND UM
WIEVIELE SEK.? 1
1 SEK. IST UM
```

Typisches Programm in „Tiny Basic". Es dient zur Verzögerung um eine bestimmte Anzahl von Sekunden

Programmzeile dafür zu sorgen, daß beim nächsten Print-Befehl oder auch bei INPUT in der gleichen Zeile weitergeschrieben wird. Ähnliches geschieht, wenn statt dem Strichpunkt ein Komma verwendet wird; dann können z. B. Zahlengruppen tabellarisch ausgedruckt werden.

Tiny Basic bietet auch die Möglichkeit, Unterprogramme in Maschinensprache aufzurufen. Dies geschieht mit dem Befehl USR, der jedoch nicht alleine stehen darf, sondern als Wertzuweisung dient, z. B. LET A = USR (256). Dieser Befehl weist der Variablen A einen Wert zu, der vom Unterprogramm, das an der Adresse 256 (dezimal) bzw. 100 (hex) steht, im Akku (LSB) und im Y-Register (MSB) in hexadezimaler Form übergeben wird. Damit sind unter anderem leistungsfähige Input-Output-Routinen realisierbar.

Verwendet man den USR-Befehl, so ist es praktisch, zu wissen, wo die Werte der Variablen gespeichert sind. Tom Pittmann hat das recht einfach gelöst: Multipliziert man das ASCII-Äquivalent des Variablennamens mit 2, so ergibt sich die Adresse, wo der entsprechende hexadezimale Wert steht. So ist die Variable A (ASCII hex 41) an den Adressen 0082 (LSB) und 0083 (MSB) gespeichert. Das vom Benutzer eingegebene Basic-Programm beginnt bei der 2000-Version an der Adresse 2900. Jede Zeile beginnt mit der hexadezimal in zwei Bytes verschlüsselten Zeilennummer; es folgt der Zeilentext in ASCII, dann die nächste Zeilennummer usw. Die Zeilentabelle ist mit zwei Null-Bytes abgeschlossen.

Zum Schluß noch eine Bemerkung über den Programmbefehl END. Er ist nur dann erforderlich, wenn das Basic-Programm keine Endlosschleife darstellt. END sorgt nämlich dafür, daß nch Ausführung des Anwenderprogramms richtig in das Tiny Basic zurückgesprungen wird. Fe.

Zeile ist nur ein Befehl möglich; der Rechenbereich umfaßt nur ganze Zahlen von –32762 bis +32762; es gibt nur 26 Variablen, nämlich A...Z, und es sind keine String-Variablen möglich. Dies führt z. B. dazu, daß die INPUT-Funktion nicht mit YES oder NO beantwortet werden kann, sondern z. B. nur mit 1 oder 0 als Antwort auf eine Frage.

Eine weitere Einschränkung ist, daß es keine FOR...NEXT-Schleifen gibt. Wie man dieses Problem umgeht, zeigt das *Bild*; es stellt einen einfachen Timer dar, der eine programmierbare Verzögerung produziert. Interessant ist dabei auch die Möglichkeit, bei Print-Befehlen mit einem Strichpunkt am Ende der

Extremwertberechnung mit TI 58/59

Zur Bestimmung der Extremwerte einer Funktion bildet man die erste Ableitung und sucht deren Nullstellen. Dieses kann bei komplizierteren Funktionen jedoch recht aufwendig werden. Das beschriebene Unterprogramm bildet näherungsweise die erste Ableitung nach der Formel

$$\frac{d}{dx} f(x) = \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Die Nullstellen werden dann mit Hilfe des Programms 08 aus dem Grundmodul (als Hauptprogramm) berechnet. Aus diesem Grund ist das Programm ziemlich kurz, was es insbesondere auch für den TI 58 (ohne Magnetkarte) geeignet erscheinen läßt. Programmzeilennummern wurden nicht angegeben, da es beliebig verschiebbar ist.

76	2nd LBL		75	–	
16	2nd A'		43	RCL	
53	(		09	09	
42	STO		11	A	
09	09		54	)	
85	+		53	(	
43	RCL		24	CE	
08	08		55	÷	
54	)		43	RCL	
53	(		08	08	
11	A		54	)	
			92	INV SBR/RTN	

Die zu untersuchende Funktion ist als Befehlsfolge in den Programmspeicher einzubringen und mit dem Label A zu versehen; die Befehle CLR, =, RST sind nicht zulässig. Der letzte Befehl muß RTN (nicht R/S) lauten. Danach wird PGM 08 aufgerufen, Untergrenze, Obergrenze, Intervallbreite sowie Fehlergrenze eingegeben und mit E gestartet. Die Zahl der noch zur Verfügung stehenden Klammerebenen wird um eine verringert, außerdem ist Speicher 09 belegt. *Ewald Bruns*

Umwandlung von Dual- in Dezimalzahlen mit dem TI 58/59

Zur Umwandlung von Dual- in Dezimalzahlen sind bereits mehrere Programme veröffentlicht worden, die aber durchweg nur bis $2^{10} = 1024$ umwandeln und recht umfangreich sind, was den Nutzwert für TI-58-Besitzer erheblich sinken läßt. Das folgende Programm erfordert für diese Umwandlung einen extrem geringen Programmieraufwand und arbeitet sehr schnell. Die Dualzahl wird dabei von links beginnend in den Rechner eingegeben, wobei die Taste A als 0 und die Taste B als 1 definiert ist. Das Umwandlungsergebnis der bisher eingegebenen Dualzahl steht jeweils in der Anzeige. Vor der Eingabe einer neuen Dualzahl ist CLR zu drücken. *Ewald Bruns*

000	76	2nd LBL	007	12	B
001	11	A	008	65	x
002	65	x	009	02	2
003	02	2	010	85	+
004	95	=	011	01	1
005	91	R/S	012	95	=
006	76	2nd LBL	013	91	R/S

Hans Werner

Mikrocomputer

Eine kleine Einführung

26. Teil

Noch'n Test...

Nicht immer wird ein Programm in der linearen Reihenfolge 0, 1, 2... abgearbeitet.

Ist das bei Punkt a, b oder c der Fall?

- ☐ a) bei einem Unterprogramm
- ☐ b) bei einem Makro
- ☐ c) weder bei a noch bei b

Bei Lösung
a) nach D
b) nach J
c) nach L

A

Die Lösung ist nicht richtig. Lesen Sie bitte die Erklärung.

```
LA  AC  k   0D      Lade die Konstante 0
AD      k   64D     Addiere 64: 0+64 = 64 (01000000)
BR  EN  d   UHU     Springe nach UHU, wenn das Ergebnis negativ ist
SH  LI      1D      Schiebe den Inhalt des Akkus um 1 nach links (10000000)
BR  EN  d   AHA     Springe nach AHA, wenn das Ergebnis negativ ist
```

Nach dem Schiebebefehl ist Bit 7 gleich 1, d. h. der Inhalt des Akkus wird vom Statusregister als negative Zahl erkannt. Somit erfolgt ein Sprung zur Adresse AHA. Natürlich ist das Bitmuster 10000000 nicht unbedingt eine negative Zahl. Es kann verschiedene Bedeutungen haben, z. B. eine Schalterstellung. Das Statusregister reagiert immer auf Bit 7, gleichgültig welche Bedeutung der Programmierer dem Inhalt des Akkus zuweist.

nach G

B

Sie haben unrecht. Die symbolische Adresse vor den Befehlen wird auch als Marke oder Label bezeichnet. In der Assemblersprache werden Marken (Adressen) eigentlich nur für Sprungziele usw. angegeben. Weitere Programmarken sind möglich, aber nicht nötig.

nach I

C

Sie haben richtig gewählt. Die nächste Aufgabe wird nicht so leicht.

nach E

D

```
        ...  312
EINS    ....
ZWEI    ....
DREI    ....
```

Für die Variablen EINS, ZWEI und DREI sollen im RAM die Speicherzellen 312...314 reserviert werden. Tragen Sie die fehlenden Angaben nach.

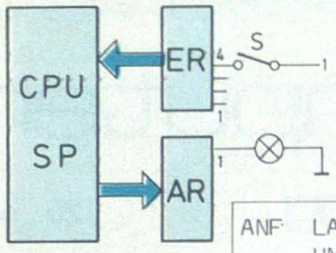
nach K

E

Sie haben richtig erkannt, daß der erste Sprungbefehl nicht ausgeführt wird. Zu diesem Zeitpunkt ist Bit 7 noch 0. Wenn Sie eine ausführliche Erklärung wünschen, lesen Sie bitte Testabschnitt B.

nach G

F



Mit dem kleinen Programm soll der Schalter S abgefragt werden. Er ist am Eingang 4 des Eingangsregisters angeschlossen. Die Lampe leuchtet nur dann, wenn der Schalter geschlossen ist. Leider war der Programmierer ein Anfänger. Er hat einige Befehle mit den falschen Adressiermerkmalen d, k, i versehen. Wie ist es richtig?

```
ANF  LA AC k  512D   Lade Schalterfeld
     UN    d    4D   Selektiere Schalter S
     BR EO d  ANF   Wenn S geöffnet: nach ANF
     LA AC d    1D  Zahl 1 laden
     SP AC d  512D  1 ausgeben (Lampe leuchtet)
     JMP   i  ANF
```

Schalter geschlossen = 1 Lampe leuchtet = 1

nach N — **G**

Sehr gut! Sie haben die richtige Antwort gefunden.

nach I — **H**

Das Sprungziel des Befehls JMP i 100 kann nicht angegeben werden, weil der Inhalt des ...

nach P — **I**

Die Antwort ist falsch. Makros werden bei der Umwandlung „Assemblersprache in Maschinencode" in das Hauptprogramm eingefügt, d. h. es gibt keine Sprünge zu einem Makroprogramm. Im Maschinencode fehlen sämtliche Makro*aufrufe*.

nach E — **J**

Lösung:

```
        ORG   312
EINS    VARI
ZWEI    VARI
DREI    VARI
```

Mit der Organisationsanweisung ORG 312 werden die Speicherplätze 312, 313 und 314 im RAM reserviert. VARI bedeutet: Variable

nach M — **K**

Das ist leider nicht richtig. Wiederholen Sie bitte die Aufgabe. (Aber nicht raten, sondern überlegen!)

nach A — **L**

```
LA AC k   0D
AD    k  64D
BR EN d  UHU
SH LI     1D
BR EN d  AHA
```

Mit welchem Befehl wird das Programm fortgesetzt?
BR EN: Springe, wenn das Ergebnis negativ ist

☐ a) mit UHU
☐ b) mit AHA

Bei Lösung
a) nach B
b) nach F

M

Lösung:

```
ANF  LA AC k  d  512D
     UN    d  k   4D
     BR EO d  d  ANF
     LA AC d  k   1D
     SP AC d  d  512D
     JMP   i  d  ANF
```

Haben Sie's gewußt?

nach O — **N**

Zum Schluß noch eine leichte Aufgabe: Was ist eine Programmarke (Label)?

☐ a) Die symbolische Adresse vor den Befehlen.
☐ b) Die Anweisung ORG vor der Variablenliste.

Bei Lösung
a) nach C
b) nach H

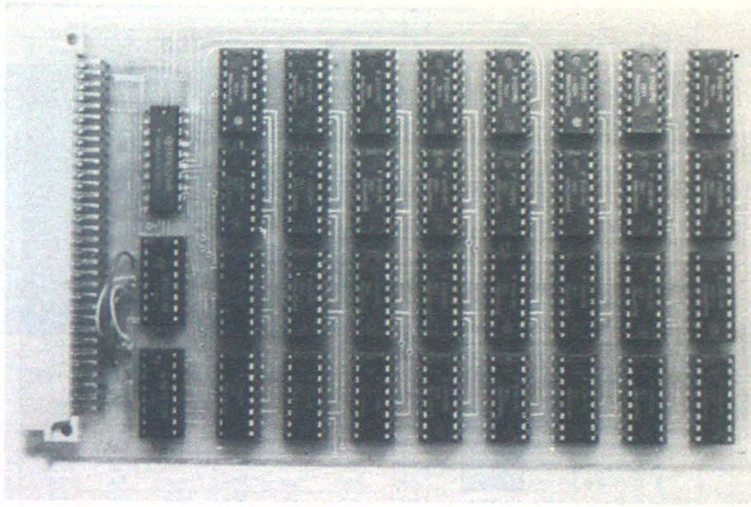
O

... weil der Inhalt des Indexregisters nicht bekannt ist. Sicherlich bedauern Sie es, daß der Test zu Ende ist. Oder etwa nicht?

PAUSE — **P**

(Fortsetzung folgt)

4-K-RAM-Karte



Auf den meisten Ein-Platinen-Mikrocomputern ist nur ein relativ geringer RAM-Speicherbereich vorgesehen. Sobald die Programme etwas umfangreicher werden, genügt dieser Speicherplatz nicht mehr. Eine preisgünstige Lösung zur Speichererweiterung stellt die beschriebene 4-KByte-RAM-Karte dar. Sie eignet sich u. a. zur Erweiterung des in der Serie „Praxis mit Mikroprozessoren" beschriebenen 8080-Systems.

Die 4K x 8 bit orientierte Speicherkarte ist an alle Mikrocomputersysteme anschließbar, die einen 8-bit-Daten- und einen 16-bit-Adressenbus besitzen. Daneben müssen nur noch zwei Steuersignale (Speicher lesen/Speicher schreiben) und +5 V zugeführt werden. Als Speicherbausteine sind die bekannten Standard-RAMs 2102 eingesetzt, die von verschiedenen Herstellern preiswert angeboten werden. *Bild 1* zeigt die Anschlußbelegung.

Das Gesamtschaltbild der Karte zeigt *Bild 2*. Jeweils acht Bausteine 2102 sind zu einem 1-K-Block zusammengefaßt. Hierin liegen alle Anschlüsse der 2102 außer den Datenein- und Ausgängen parallel.

Die acht Dateneingangsleitungen eines Blocks (jeweils Pin 11 eines 2102) sind über den Schnittstellenstecker direkt an den Datenbus des Rechners gelegt, wie die Dateneingangsleitungen der übrigen drei 1-K-Blocks. Ebenso führen die zehn Adreßleitungen aller Blocks (A0...A9) über den Stecker direkt an den Adreßbus des Rechners.

Dies kann ungepuffert geschehen, da der Eingangsstrom eines 2102-Einganges maximal 10 µA beträgt, der Adreß- und Datenbus also mit maximal 40 µA belastet werden. Die Datenausgänge aller Blocks (jeweils Pin 12 am 2102) führen an die Eingänge des Achtfach-Treiberbausteines 74244. Sie werden nur dann verstärkt auf den Rechner-Datenbus gelegt, wenn dieser Baustein ein Chip-Select-Signal erhält.

Mit 10 Adreßleitungen lassen sich 2^{10}, also 1024 Byte adressieren. Welcher der vier Blöcke nun angesprochen wird, wird durch die zwei höherwertigeren Adreßbits A10 und A11 bestimmt. Diese führen an die Eingänge A0 und A1 des 1-aus-8-binär-Decoders 74138.

Abhängig von den Zuständen an den Eingängen A0 und A1 wird einer der vier beschalteten Ausgänge zu log. Null werden. Jeder dieser Ausgänge führt auf die $\overline{CE}$-Eingänge jeweils eines 1-K-Blockes, jeweils ein Block wird also freigegeben. Wo nun der 4-K-Bereich innerhalb des gesamten 64-K-Bereiches liegt, wird durch die restlichen Adreßbits A12...A15 bestimmt. Diese Decodierung wir ebenfalls vom IC 74138 durchgeführt. Dabei müssen die Freigabeeingänge E1 und E2 „Ø", E3 „1" sein. Weiter muß der Eingang A2 „Ø" sein, da nur die Ausgänge O 0...3 beschaltet sind. Soll nun der Adreßbereich der Karte etwa ab der hexadezimalen Adresse 1000 liegen, müssen die oberen vier Adreßleitungen, die dann den Zustand

A15 = 0 A14 = 0 A13 = 0 A12 = 1

annehmen, entsprechend mit diesen Eingängen verbunden werden, also A12 auf E3, A13 bis A15 auf $\overline{E2}$, $\overline{E1}$ und A2.

Bei Adreßbereichen, bei denen mehr als ein Adreßbit auf „1" liegt, etwa 3000 als Startadresse, müssen diese Adreßbits UND-verknüpft auf E3 geführt werden. Dies geschieht durch eine Dioden-Widerstandskombination. Eine Gesamtübersicht der Festlegung des Adreßbereiches zeigt *Bild 3*.

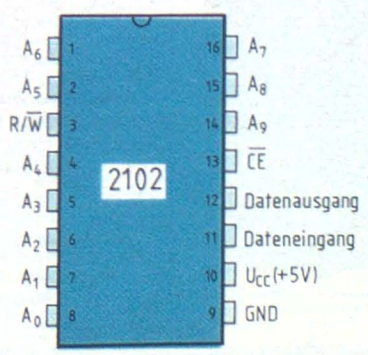


Bild 1. Anschlußbelegung der verwendeten RAM-Chips 2102

Durch diese etwas „holprige" Art der Adreßbereichsfestlegung unterscheidet sich die 4-K-Karte von professionellen Ausführungen, die dazu meist einen DIL-Schalter mit entsprechender Logik verwenden. Dies ist jedoch hier nicht nötig, da beim Hobbyelektroniker der einmal gewählte Adreßbereich meist fest bleibt.

Aus Bild 3 erkennt man ferner, daß der Adreßbereich 0000...0FFF nicht ansprechbar ist. Auch dies ist kein großer Nachteil, da ab Adresse 0000 in jedem System entweder Monitorprogramm (8080, Z 80) oder ein schon in der Grundversion vorhandenes RAM steht (6800, 6502).

Die hohe Packungsdichte zwang auch zu einer unkonventionellen Lösung der Durchschaltungsfreigabe der Datenausgänge auf den Rechner-Datenbus. Die Durchschaltung wird mit dem 74244 erledigt, wie schon oben beschrieben wurde. Die Ausgänge eines Blocks dürfen nur dann auf den Datenbus gelegt werden, wenn ein Block adressiert ist und das Steuersignal $\overline{MEMR}$ (Speicher lesen) vom Rechner auf Null gelegt wird. Diese Verknüpfung führt der 74148, ein binärer 8-zu-3-Encoder durch. Sein Ausgang GS geht nur dann auf Null, wenn der Eingang E1 und mindestens einer der Eingänge 0...3 auf Null liegen. GS schaltet dann über die CS-Eingänge den Bustreiber 74244 durch.

Das Steuersignal $\overline{MEMW}$ (Speicher lesen) führt direkt an die R/$\overline{W}$-Eingänge

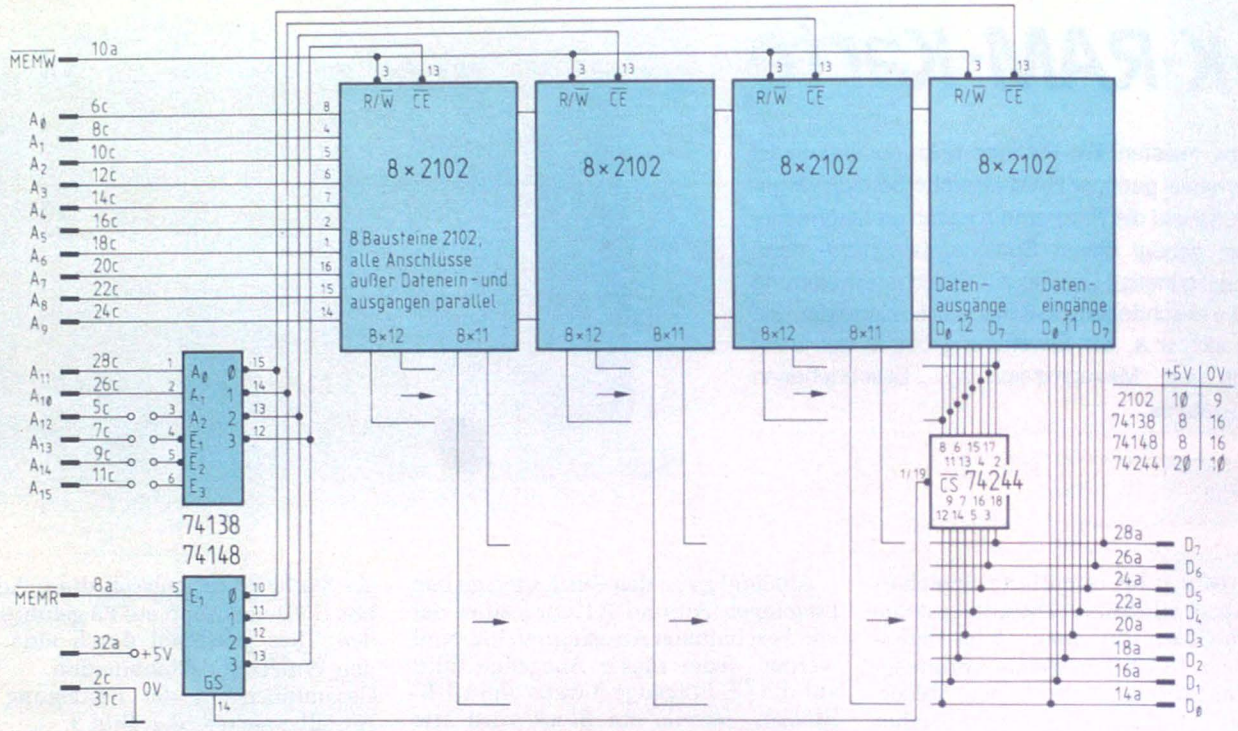


Bild 2. Gesamtschaltbild der 4-K-Speicherkarte. Die angegebene Steckerbelegung ist kompatibel zum Siemens-SMP-80-System

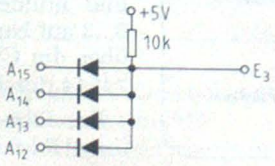
aller 2102-Speicherbausteine. Ist eine Zelle des 4-K-Bereiches adressiert und MEMW aktiv, so wird die auf dem Datenbus befindliche Bitkombination gespeichert.

Eine Bausatzversion sowie ein geprüftes Fertiggerät sind bei der Firma Graf Elektronik Systeme GmbH (Postfach 1610, 8960 Kempten) erhältlich.

Bild 4 zeigt die Unterseite der doppelseitig-durchkontaktierten Platine mit den (leider) notwendigen Drahtbrücken. *Bild* 5 zeigt die Bestückungsseite, wobei Pin 1 der ICs jeweils durch einen Punkt gekennzeichnet ist. Daraus geht auch hervor, wie die ICs den Datenbits zugeordnet sind, was die Fehlersuche sehr vereinfacht. Speziell bei Verwendung preiswerter RAM-ICs ist es vorteilhaft, Sockel zu verwenden.

Bereich	A_2 (Pin 3)	$\overline{E_1}$ (Pin 4	$\overline{E_2}$ (Pin 5	E_3 (Pin 6)
1000 – 1FFF	A_{15}	A_{13}	A_{14}	A_{12}
2000 – 2FFF	A_{12}	A_{15}	A_{14}	A_{13}
3000 – 3FFF	A_{15}	M	A_{14}	$A_{12} \wedge A_{13}$
4000 – 4FFF	A_{12}	A_{13}	A_{15}	A_{14}
5000 – 5FFF	A_{15}	M	A_{13}	$A_{12} \wedge A_{14}$
6000 – 6FFF	A_{15}	M	A_{12}	$A_{13} \wedge A_{14}$
7000 – 7FFF	M	M	A_{15}	$A_{12} \wedge A_{13} \wedge A_{14}$
8000 – 8FFF	A_{12}	A_{13}	A_{14}	A_{15}
9000 – 9FFF	A_{13}	M	A_{14}	$A_{12} \wedge A_{15}$
A000 – AFFF	A_{12}	M	A_{14}	$A_{13} \wedge A_{15}$
B000 – BFFF	M	M	A_{14}	$A_{12} \wedge A_{13} \wedge A_{15}$
C000 – CFFF	A_{12}	M	A_{13}	$A_{14} \wedge A_{15}$
D000 – DFFF	M	M	A_{13}	$A_{12} \wedge A_{14} \wedge A_{15}$
E000 – EFFF	M	M	A_{12}	$A_{13} \wedge A_{14} \wedge A_{15}$
F000 – FFFF	M	M	M	$A_{12} \wedge A_{13} \wedge A_{14} \wedge A_{15}$

Bild 3. Festlegung des Adressenbereichs der RAM-Karte durch UND-Verknüpfung der Adreßleitungen



Beispiel: UND-Verknüpfung (für den Adreßbereich F000 – FFFF)

Speicher-Test

Haben Sie alles zusammengebaut? Das untenstehende 8080-Maschinencode-Programm testet den Speicher, indem es in jede Zelle alle Werte von 00 bis FF einträgt. Die Anfangsadresse des zu testenden RAM muß im HL-Register stehen, die Endadresse im DE-Register. Wenn alle Zellen funktionieren, erfolgt die Rückkehr mit gelöschtem Carry-Bit, andernfalls wird es gesetzt und die Fehleradresse im HL-Register übergeben. Das Programm ist beliebig verschiebbar.

```
0000 C5          0012 C20400
0001 0E00        0015 0C
0003 E5          0016 CA1D00
0004 71          0019 E1
0005 7E          001A C30300
0006 B9          001D AF
0007 C22100      001E E1
000A 23          001F C1
000B 7B          0020 C9
000C BD          0021 37
000D C20400      0022 C1
0010 7A          0023 C1
0011 BC          0024 C9
```

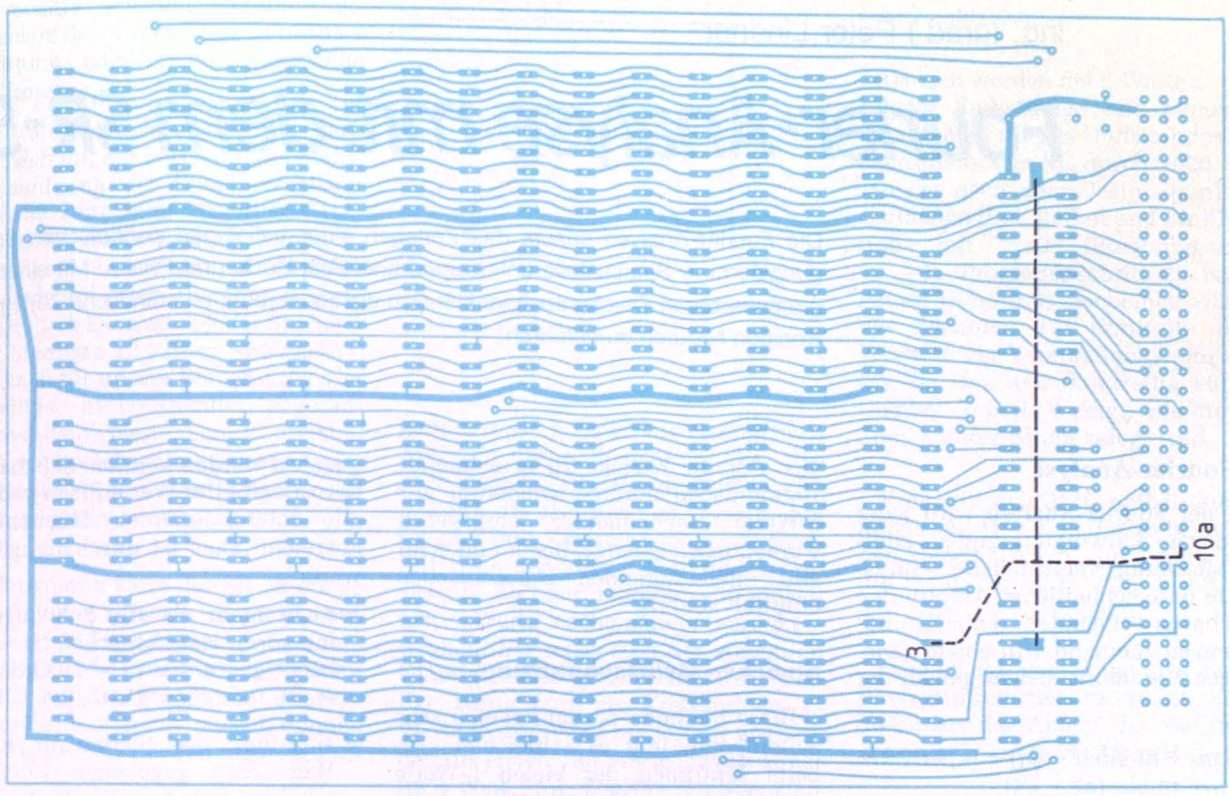


Bild 4. Auf der Lötseite der Platinen sind zwei Drahtbrücken erforderlich

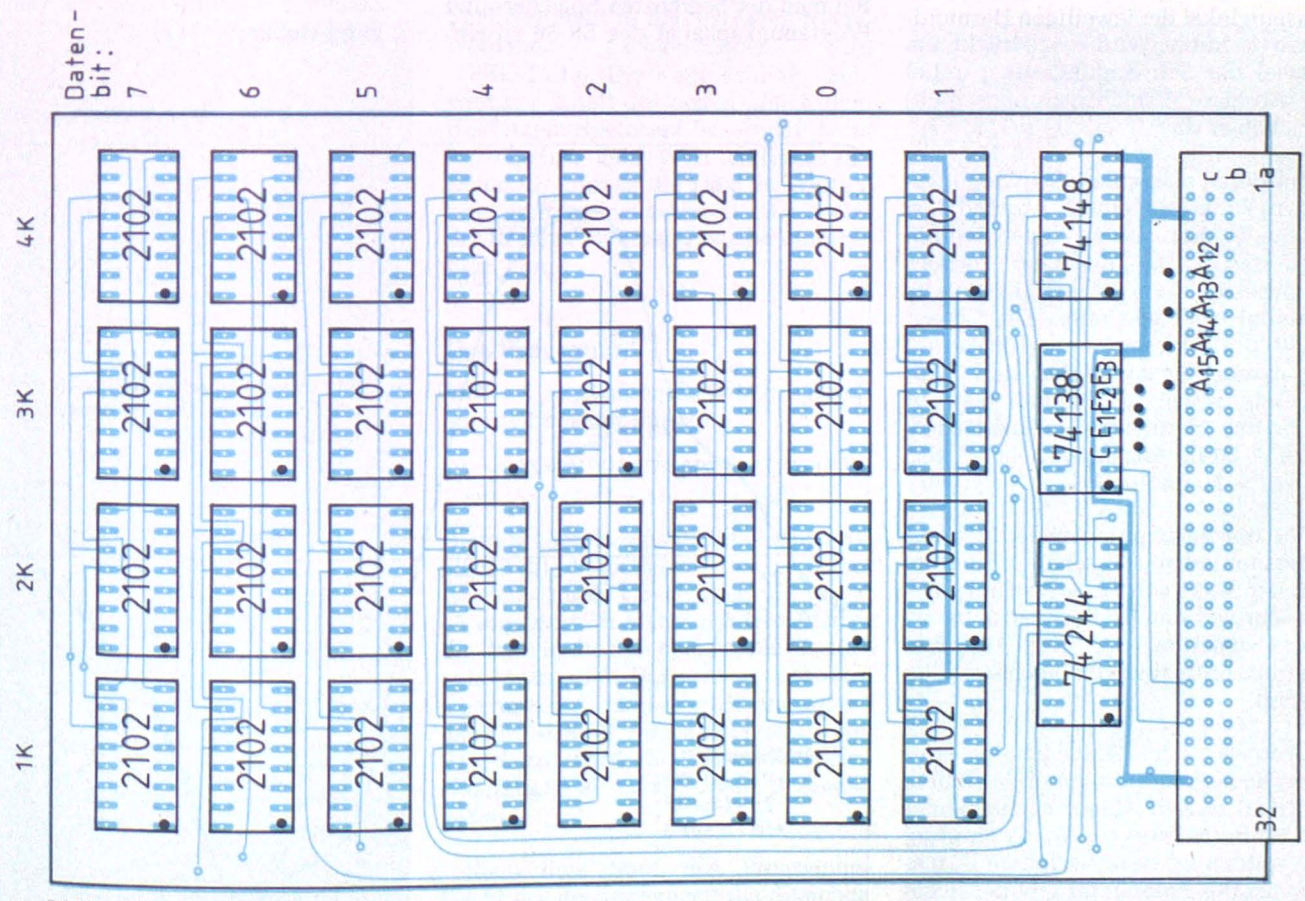


Bild 5. Bestückungsplan der 4-KByte-Karte

Ing. (grad.) Peter Lindner

Fourier-Analyse mit dem SR-56

Die Fourier-Analyse erlaubt das Zerlegen einer beliebigen periodischen Schwingungsform in Sinusanteile unterschiedlicher Amplitude und Phase. Mit einem programmierbaren Taschenrechner läßt sich die sonst recht umständliche Zerlegung in wenigen Minuten durchführen.

Die Fourier-Analyse

Bei der Fourier-Analyse wird jeder periodische Schwingungszug als additive Mischung verschiedener Sinus-Anteile unterschiedlicher Amplituden und Phasen verstanden, deren Grundfrequenzen zueinander in einem ganzzähligen Verhältnis stehen gemäß der Form:

$$f(t) = m_o + m_1 \sin(t + \varphi_1) + m_2 \sin(2t + \varphi_2) + \ldots + m_n \sin(nt + \varphi_n)$$

Dabei sind m die Amplituden und φ die Phasenwinkel der jeweiligen Harmonischen [1]. Mit m_o wird ausgedrückt, um wieviel der Schwingungszug parallel zur Abszisse (Gleichspannungsanteil) verschoben ist.

Entnimmt man einem Schwingungszug n Punkte in einem Abstand $t = 360°/n$, so kann man aus deren Ordinatenwerten $f_{i1} \ldots f_{in}$ mit einer Fourier-Analyse die Werte für m und φ für alle Harmonischen gewinnen. Die Genauigkeit des Ablesens und die Anzahl der entnommenen Punkte sind maßgebend, wobei n Punkte bis zur (n/2–1)ten Harmonischen ausreichen; bei n = 20 im folgenden Beispiel also bis zur 9. Harmonischen.

Für die Berechnung bietet sich ein programmierbarer Taschenrechner an, z. B. der SR-56 von Texas Instruments – ein sehr verbreiteter Rechner der vorigen Generation, der über 100 Programmschritte und 10 Speicherregister verfügt.

Grundsätzlich läßt sich für alle derartigen Rechner ein Programm erstellen, während dessen Ablauf nacheinander die n Ordinatenwerte $f_{i1} \ldots f_{in}$ eingegeben werden müssen. Nach dem Durchrechnen des Programms gibt der Rechner Amplitude m_H und Phase φ_H der jeweils eingesetzten Harmonischen H aus. Für die nächste zu berechnende Harmonische müßten wiederum alle f_i-Werte nacheinander eingegeben werden usw., solange, bis m und φ für alle interessierenden Harmonischen ermittelt sind [2].

Speicher-Mehrfachnutzung

Diese Methode ist naturgemäß sehr zeitraubend, und es schleichen sich beim Eintippen der vielen f_i-Werte leicht Fehler ein. Deshalb stellt sich die Frage, ob sämtliche Werte für f_i im Rahmen der begrenzten Speicher- und Programmkapazität des SR-56 so eingegeben werden können, daß dies zum Berechnen aller Werte für m und φ für alle interessierenden Harmonischen ausreicht. Dies ist durchaus möglich, wenn

1. nicht mehr als alle Schwingungen bis zur 9. Harmonischen (H = 1...9) benötigt werden (n = 20 Ordinatenwerte f_i reichen dann aus);
2. sich max. drei Stellen für jeden f_i-Wert dem gegebenen Kurvenzug entnehmen lassen, alle Werte also in der Form „ZE,z" oder „E,zh" (Z = Zehner, z = Zehntel usw.) zur Verfügung stehen;

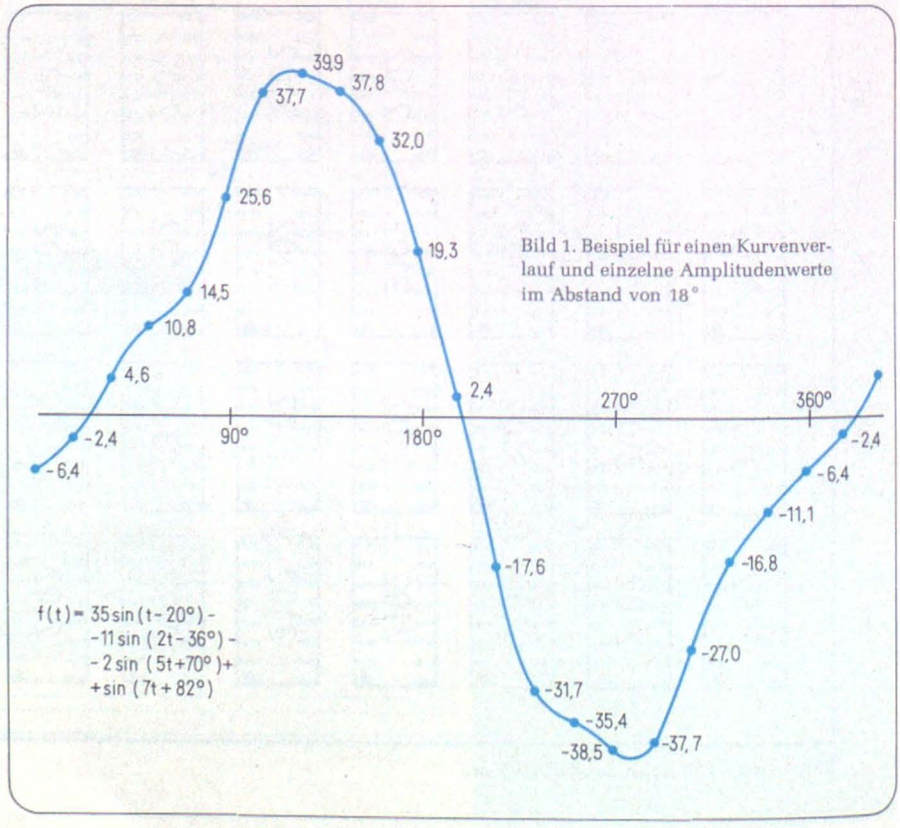


Bild 1. Beispiel für einen Kurvenverlauf und einzelne Amplitudenwerte im Abstand von 18°

3. alle f_i-Werte durch Vergrößern von m_o positiv gemacht werden (Verschieben des Kurvenzugs in positive Richtung parallel zur gegebenen Nullinie).

Daß f_i hierbei nicht negativ sein darf, erklärt sich mit der notwendigen Mehrfachausnutzung der Speicherregister: Der SR-56 kann eine 10stellige Mantisse und einen zweistelligen Exponenten anzeigen – intern rechnet der Rechner jedoch mit einer 12stelligen Mantisse. Deshalb kann auch jeder Speicher in der Mantisse 12 Ziffern speichern – sinngemäß für unsere Berechnung vier dreistellige, hintereinander angeordnete positive f_i-Werte. Um 20 dreistellige Werte speichern zu können, würden demnach fünf der verfügbaren zehn Speicherregister benötigt.

Im Programm kann der SR-56 die zu verarbeitenden f_i-Werte immer nur aus einem vorher festzulegenden Speicher (im Beispiel M_6) entnehmen, weil andernfalls mit den gegebenen 100 Programmschritten nicht auszukommen wäre. Also erhält der Rechner im Programm zusätzlich die Anweisung, nach Ableeren dieses Speicherinhalts, also nach dem Verarbeiten der ersten vier f_i-Werte, die nächsten vier Werte in dieses nun freie Register nachzuschieben. Mit Hilfe der Integer-Funktion lassen sich im Rechnerprogramm die einzelnen f_i-Werte nacheinander zurückgewinnen. Bei sechs statt der eigentlich benötigten fünf Speicher für 20 dreistellige Werte kann ein Kreislauf der f_i-Werte in der Weise hergestellt werden, daß die ersten vier Werte jeweils außer im 6. noch im 1. Register der Speicherkette enthalten sind, wobei während der Verarbeitung jeweils ein voller Speicherinhalt in den nächsten freigewordenen nachgeschoben wird (*EXC-Taste). Die Kette schließt sich, weil der Inhalt des ersten gleich dem des letzten Speichers ist, so daß beim Verschieben kein Wert verlorengeht. Demnach müssen die max. 20 Werte f_i nur einmal eingegeben werden, weil der Rechner sie immer wieder automatisch abruft.

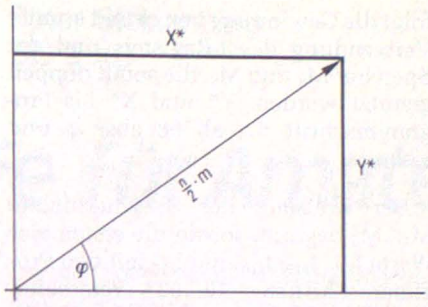


Bild 2. Zusammenhang zwischen rechtwinkligen und Polarkoordinaten

Tabelle 1.
Amplitudenwerte zu Bild 1

i	f_i	f_i +45	aufzunehmen in Speicher
1	–6,4	38,6	
2	–2,4	42,6	M_2
3	4,6	49,6	
4	10,8	55,8	
5	14,5	59,5	
6	25,6	70,6	M_3
7	37,7	82,7	
8	39,9	84,9	
9	37,8	82,8	
10	32,0	77,0	M_4
11	19,3	64,3	
12	2,4	47,4	
13	–17,6	27,4	
14	–31,7	13,3	M_5
15	–35,4	9,6	
16	–38,5	6,5	
17	–37,7	7,3	
18	–27,0	18,0	M_6 und M_1
19	–16,8	28,2	
20	–11,1	33,9	
1	–6,4	38,6	
2	–2,4	42,6	

Das Programm

Bild 1 zeigt einen Schwingungszug, ermittelt aus der Funktion

$$f(t) = 35 \sin (t–20°) – 11 \sin (2t–36°) 2 \sin (5t + 70°) + \sin (7t + 82°).$$

Hieraus können die 20 Ordinatenwerte gemäß Bild 1 und *Tabelle 1* entnommen werden. Durch Verschieben um m_o = 45 ergeben sich lauter positive Werte, die dann in die Speicher M_1 bis M_6 aufzunehmen sind. Das geht am schnellsten mit dem kleinen Hilfsprogramm gemäß *Tabelle 2*. Sind alle Werte in den Speichern, wird das eigentliche Rechenprogramm nach *Tabelle 3* eingegeben.

Nach dem Start des Programms löscht der Rechner zuerst die Speicher M_8 und M_9 sowie das t-Register und hält die Anzahl der Kurvenpunkte n = 20 im Register M_o (= i_1) fest (Programmschritte 00...09).

Danach werden die f_i-Werte in umgekehrter Reihenfolge wie eingegeben entnommen mittels Division des Speicherinhalts von M_6 durch 1000, Ablösen der nach dem Dezimalpunkt entstandenen drei Stellen und Multiplikation mit 100 (Programmschritte 10...29). Programmschritt 29 ist eine Pause: Hierdurch erscheinen während des Rechnens zur Kontrolle alle f_i-Werte in der Verarbeitungsreihenfolge f_{i20}, f_{i19}, f_{i18} usw. kurzzeitig auf dem Display. *Tabelle 4* zeigt, wie die zehn Speicher des SR-56 belegt sind.

Programmschritt 92 enthält die Ordnungszahl H = 1...9 der zu berechnenden Harmonischen, die jeweils durch die Tastenfolge GTO / 9 / 2 / LRN ausgewechselt werden kann.

Wegen der Verzweigungen bei den Programmschritten 12, 30, 40, 50 und 80 dauert der Ablauf des Rechenprogramms für einen Wert von H ca. 90 Se-

Tabelle 2. Hilfsprogramm zum Abspeichern der Wertetabelle

Programm-Schritt	Code	Taste	Bemerkung
00	64	x	Eingabe 1.
01	01	1	f_i-Wert
02	44	EE	
03	07	7	
04	84	+	
05	41	R/S	Eingabe 2.
06	64	x	f_i-Wert
07	04	4	
08	19	$*10^x$	
09	84	+	
10	41	R/S	Eingabe 3.
11	64	x	f_i-Wert
12	01	1	
13	00	0	
14	94	=	
15	64	x	
16	03	3	
17	19	$*10^x$	
18	84	+	
19	41	R/S	Eingabe 4.
20	64	x	f_i-Wert
21	01	1	
22	00	0	
23	94	=	
24	41	R/S	Ausgabe zur
25	42	RST	Übernahme in Speicher

kunden. Danach erscheint im Display die Amplitude m_H und nach Drücken der Taste x≷t der Phasenwinkel φ_H der im Programmschritt 92 enthaltenen Harmonischen H.

Mit dem durch die Schritte 30...32 und 40...42 aufgerufenen Unterprogramm 83...94 beginnt die eigentliche Fourier-Analyse durch Bilden des Produkts $(i-1) \cdot t \cdot H$, wobei $t = 360°/n = 18°$ ist. Von diesem Produkt wird im Programm der Sinus und der Cosinus errechnet und mit dem jeweiligen f_i-Wert multipliziert. Da diese Produkte in die Speicher M_8 und M_9 addiert werden, enthält ab Programmschritt 50:

M_9 den Wert

$$X^* = \sum_{i=1}^{20} f_i \cdot \sin[(i-1) \cdot t \cdot H]$$

und M_8 den Wert

$$Y^* = \sum_{i=1}^{20} f_i \cdot \cos[(i-1) \cdot t \cdot H]$$

Bild 2 zeigt den Zusammenhang zwischen X*, Y*, m und φ.

Bei der Umwandlung von rechtwinkligen in Polarkoordinaten im SR-56 erfolgt die Gewinnung von m und φ unter Verwendung des t-Registers und der Speicher M_8 und M_9, die somit doppelt genutzt werden: Y* und X* bis Programmschritt 57, ab 58 aber φ und $n/2 \cdot m$.

Der Kreislauf der Speicherinhalte $M_1...M_6$ beginnt, sobald die ersten vier Werte f_{i20}, f_{i19}, f_{i18} und f_{i17} mit den Programmschritten 10...52 verarbeitet sind. Der Inhalt von M_6 ist dann Null, und der Verzweigungsbefehl der Schritte 12...14 löst das Verschieben der Speicherinhalte während der Programmschritte 66...82 aus. Danach ist M_6 wieder mit den nächstfolgenden vier Werten $f_{i16}...f_{i13}$ gefüllt, und der Ablauf kann von neuem beginnen. Wenn alle 20 f_i-Werte verarbeitet, d. h. jeweils mit dem Sinus und dem Cosinus von $(i-1) \cdot t \cdot H$ multipliziert und in die Speicher M_8 und M_9 aufaddiert sind, ist auch die im Speicherregister M_o zu Beginn des Programms festgehaltene Zahl n = 20 = i durch den Befehl *dsz (50...52) zu Null geworden. Der *dsz-Befehl zieht bei jedem Durchlauf automatisch 1 vom Inhalt des Speichers M_o ab. Solange $M_o > 0$ ist, springt das Programm von Schritt 52 auf Schritt 10; ist $M_o = 0$, geht es mit Schritt 53 weiter, bis im Display der Wert für m_H erscheint und nach Drücken der Taste x≷t der Wert für φ_H.

Tabelle 5 zeigt die errechneten Werte für m und φ für die Harmonischen $H_1...H_9$. Der Rechner zeigt m immer positiv an; aber m ist dann negativ, wenn der angezeigte Phasenwinkel $\varphi > 90°$ ist. Wird φ negativ oder $90° < \varphi < 180°$ angezeigt, ist auch φ negativ. Wird $\varphi < 90°$ oder $\varphi > 180°$ angezeigt, ist φ positiv.

Wie ein Vergleich mit der Ausgangsfunktion zeigt, sind die Abweichungen von φ gering, aber unvermeidbar wegen der gerundeten Werte der Kurvenpunkte. Die für die nicht enthaltenen Harmonischen 3, 6, 8 und 9 ausgewiesenen Werte für φ sind ohne Bedeutung, da m ohnehin Null ist. Sinngemäß kann das Programm für 30 zweistellige f_i-Werte statt der berechneten 20 dreistelligen Werte modifiziert werden.

Literatur

[1] *Tönnies, Chr.:* Praktische und angewandte Mathematik, Verlag W. Girardet, Essen

[2] *Wilcox, Alan D.:* „Program analyzes spectrum of oscilloscope wave forms." Electronics, Februar 1977

Tabelle 3. Programm zur Berechnung der Sinuskomponenten (Fourier-Analyse)

Programm-Schritt	Code	Taste	Programm-Schritt	Code	Taste	Programm-Schritt	Code	Taste	Programm-Schritt	Code	Taste
00	00	0	25	19	*10^x	50	27	*dsz	75	03	3
01	33	STO	26	94	=	51	01	1	76	39	*EXC
02	08	8	27	33	STO	52	00	0	77	04	4
03	33	STO	28	07	7	53	34	RCL	78	39	*EXC
04	09	9	29	59	*pause	54	09	9	79	05	5
05	56	*CP	30	57	*subr	55	32	x≷t	80	22	GTO
06	02	2	31	08	8	56	34	RCL	81	01	1
07	00	0	32	03	3	57	08	8	82	00	0
08	33	STO	33	24	cos	58	26	*f(n)	83	34	RCL
09	00	0	34	64	x	59	03	R→P	84	00	0
10	34	RCL	35	34	RCL	60	32	x≷t	85	74	–
11	06	6	36	07	7	61	54	:	86	01	1
12	37	*x=t	37	94	=	62	01	1	87	94	=
13	06	6	38	35	SUM	63	00	0	88	64	x
14	06	6	39	08	8	64	94	=	89	01	1
15	54	:	40	57	*subr	65	41	R/S	90	08	8
16	03	3	41	08	8	66	34	RCL	91	64	x
17	19	*10^x	42	03	3	67	05	5	92	01	1
18	74	–	43	23	sin	68	33	STO	93	94	=
19	29	*Int	44	64	x	69	06	6	94	58	*rtn
20	33	STO	45	34	RCL	70	39	*EXC			
21	06	6	46	07	7	71	01	1			
22	94	=	47	94	=	72	39	*EXC			
23	64	x	48	35	SUM	73	02	2			
24	02	2	49	09	9	74	39	*EXC	Anzeigeformat: *fix/1		

Tabelle 4. Inhalt der Speicherregister

M	Inhalt
0	i = 20, 19, 18 ...
1	$f_{i17}...f_{i20}$
2	$f_{i1}...f_{i4}$
3	$f_{i5}...f_{i8}$
4	$f_{i9}...f_{i12}$
5	$f_{i13}...f_{i16}$
6	$f_{i17}...f_{i20}$
7	f_i-Zwischenspeicher
8 y*	bis Schritt 57 / φ / n/2 · m / ab Schritt 58
9 x*	

Tabelle 5. Die aus Bild 1 errechneten Ergebnisse

H	m	φ
1	35,0	– 20,0
2	11,0	144,1
3	0,0	(– 4,8)
4	0,0	(180,0)
5	2,0	249,6
6	0,0	(– 32,1)
7	1,0	82,5
8	0,0	(– 0,0)
9	0,0	(38,5)

Hans Werner

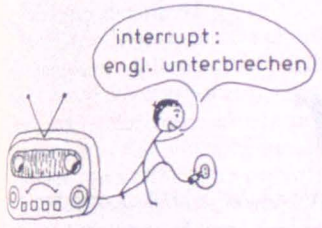
Mikrocomputer

Eine kleine Einführung

27. Teil

Interrupt

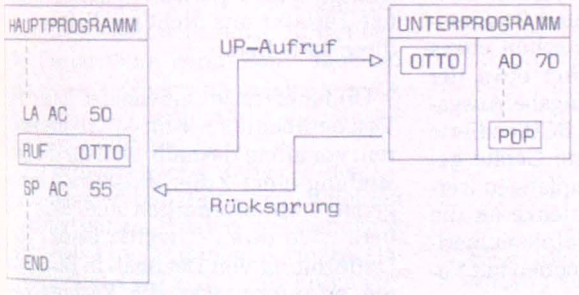
Immer wieder ist der geheimnisumwitterte Begriff INTERRUPT aufgetaucht. Wir erinnern uns: Der Interrupteingang war der dritte Steuereingang der CPU. Wird er mit einem Impuls aktiviert, unterbricht die CPU das laufende Programm und setzt es an anderer Stelle fort. Es wird in das soganannte Interruptprogramm gesprungen.



Soll das Hauptprogramm wieder aufgenommen werden, muß der letzte Befehl des Interruptprogramms POP sein.

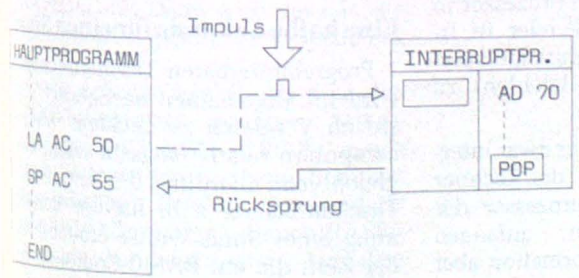
Interessant ist der Vergleich zwischen Unterprogramm und Interrupt.

Unterprogrammaufruf:



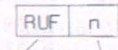
Der Befehl RUF veranlaßt die CPU, das Unterprogramm OTTO zu bearbeiten. OTTO ist die Adresse bzw. der Name des Unterprogramms. Mit dem Befehl POP erfolgt der Rücksprung.

Interruptrutine:

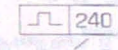


Während die CPU z. B. den Befehl LA AC 50 bearbeitet, erhält sie ein Interruptsignal (Impuls). Sie führt noch den Befehl vollständig aus und arbeitet dann mit dem Interruptprogramm weiter. Der Befehl POP veranlaßt den Rücksprung in das Hauptprogramm.

Unterprogramm:
Aufruf durch den Befehl



Interruptprogramm:
Aufruf durch einen Impuls



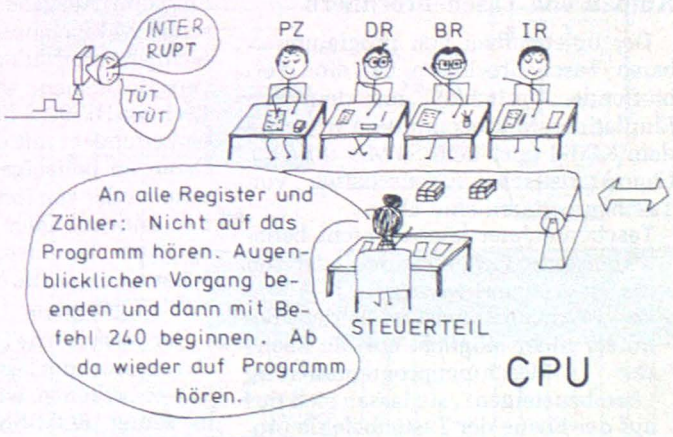
Die Interruptaufforderung ist kein Befehl. Im Gegensatz zum Unterprogramm ist der Interrupt nicht im Programm enthalten, sondern kann jederzeit von außen ausgelöst werden, also per Handware.
Das Programm wird sofort unterbrochen, nachdem der gerade bearbeitete Befehl beendet ist.

Eine Angabe der Interrupt-Sprungadresse ist nicht möglich und auch nicht nötig. Sie wird vom Hersteller festgelegt. Für unser Modell wählen wir 240, d. h. jeder Impuls am Interrupteingang bewirkt einen Programmsprung nach Adresse 240 (Speicherzelle im ROM).

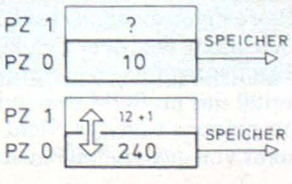
> Die Interruptaufforderung ist eine Vorrang- bzw. Prioritätsunterbrechung. Durch ein externes Signal wird der Prozessor aufgefordert, im Hauptprogramm anzuhalten und eine bestimmte andere Programmstelle zu bearbeiten. Anschließend kann das Hauptprogramm wieder aufgenommen werden, als hätte es keine Unterbrechung gegeben.

Interrupt-Ablauf

Die interne Steuerung von Unterprogrammaufruf (Punkt B58) und Interruptanforderung ist fast identisch. Ein Beispiel: PZ 0 adressiert Speicherzelle 10. In diesem Moment kommt der Interruptimpuls. Befehl 10 wird noch ausgeführt. Ist er z. B. ein 3-Byte-Befehl, steht PZ 0 dann auf 12.



Jetzt übernimmt PZ 1 den Inhalt von PZ 0 und inkrementiert auf 13. PZ 0



wird automatisch mit Adresse 240 geladen.
Steht am Schluß des Interruptprogramms der Befehl POP, läuft der umgekehrte Vorgang ab. Das Hauptprogramm wird wieder mit der Folgeadresse 13 aufgenommen.

Initialisierung

Unser Modell hat kein richtiges Stackregister, sondern nur die beiden Zähler PZ 0 und PZ 1. Mit dieser Anordnung ist nicht mehr als eine Rücksprungadresse möglich.

Kommt während eines Unterprogramms ein Interruptsignal, geht die Rücksprungadresse des Unterprogramms verloren, und das Hauptprogramm kann nicht wieder aufgenommen werden. In vielen Fällen ist das nicht erwünscht. Deshalb kann mit einem besonderen Befehl der Interrupt erlaubt bzw. verboten werden.

Setze das Interrupt-Flip-Flop	SET I	n

Das Interrupt-Flip-Flop ist im Statusregister SR enthalten (siehe Punkt B4). n kann entweder 0 oder 1 sein.

n=0: Das Interruptsignal hat keine Wirkung

n=1: Das Interruptsignal löst einen Interrupt aus.

(Schluß folgt)

```
Beispiel: SET I  0D    vor UP kein Interrupt möglich
          RUF    ...   UP-Aufruf
          SET I  1D    Interrupt möglich
```

Man sagt, der Interrupt muß initialisiert werden.

Mikrocomputer contra Taschenrechner

Programmierbare Taschenrechner werden oft als „Vorstufe" zu Mikrocomputern angesehen. In Wahrheit sind sie vollständige Mikrocomputer, meist aus einem 4-bit-Prozessor, einem Betriebssystem im ROM und einem RAM zum Speichern von Zahlen und Programmschritten bestehend. Der folgende Beitrag vergleicht übliche Mikrocomputer mit programmierbaren Taschenrechnern.

Aufbau von Taschenrechnern

Der Innenaufbau von programmierbaren Taschenrechnern hat eine verblüffende Ähnlichkeit mit dem von Einplatinen-Mikrocomputern wie etwa dem KIM-1 oder dem SYM-1 (*Bild 1*). Charakteristische Eigenschaften von Taschenrechnern sind aber:

- Taschenrechner können nicht beliebige externe Geräte steuern oder von diesen gesteuert werden.
- Bei Taschenrechnern ist es dem Benutzer nicht möglich, auf die Ebene der Maschinenprogrammierung „herabzusteigen", sie lassen sich nur auf der Ebene der Tastenbefehle programmieren und bedienen.
- Eine alphanumerische Ausgabe ist – wenn überhaupt – nur mit einem Drucker möglich; eine Textverarbeitung im üblichen Sinne ist nicht möglich.

Diese Eigenschaften resultieren eigentlich nicht aus dem Prinzipaufbau der Taschenrechner, sondern sind ein Resultat des im ROM stehenden Betriebsprogramms und des Nicht-Vorhandenseins von geeigneten Bausteinen für die Ein- und Ausgabe. Die Grenzen zwischen Mikrocomputern und Taschenrechnern sind allerdings schon etwas verschwommen; so verfügt etwa der Rechner HP-97 S über Eingabe/Ausgabe-Leitungen, mit denen BCD-codierte Daten an beliebige andere Geräte gesendet oder von ihnen empfangen werden können. Oder man denke an die neuesten Rechner mit alphanumerischer Anzeige, die das Rechnen mit Variablen gestatten.

Der wesentliche Unterschied ist also: Taschenrechner lassen sich ausschließlich so bedienen, wie der Hersteller das in seiner ROM-Software vorgesehen hat; es ist – von einigen völlig uneffizienten „Tricks" abgesehen – nicht möglich, den eingebauten Prozessor in seiner Maschinensprache oder in irgendeiner anderen Programmiersprache (außer den Tasten-Befehlen) zu programmieren.

Aus diesem Grunde ist es zwar interessant zu hören, daß z. B. der Rechner TI-59 mit einem Mikroprozessor der TMS-1000-Serie arbeitet, anfangen kann man mit dieser Information aber wenig, denn man kann sozusagen von der Tastatur aus nicht per Software an diese CPU heran.

Üblicherweise verwendet man für Taschenrechner 4-bit-Mikroprozessoren; vor allem deshalb, weil für die Darstellung einer Ziffer zwischen 0 und 9 genau 4 bit erforderlich sind. Eine größere „Wortbreite" würde bereits eine Umformung von Dezimal- in Binärzahlen erfordern, was die Rechner-Software für den Hersteller kompliziert und u. U. auch die Verarbeitungsgeschwindigkeit herabsetzt. Taschenrechner arbeiten intern also im BCD-Format. *Bild 2* zeigt die Innereien des programmierbaren Rechners HP-25.

Eine halbe Sekunde für einen Sinus

Programmierbaren Taschenrechnern wird im allgemeinen nachgesagt, daß sie im Vergleich zu „echten" Mikrocomputern relativ langsam seien. Vergleicht man allerdings die Zeit, die ein Taschenrechner z. B. für die Berechnung eines Sinus-Wertes braucht, mit der Zeit, die ein BASIC-Computer für

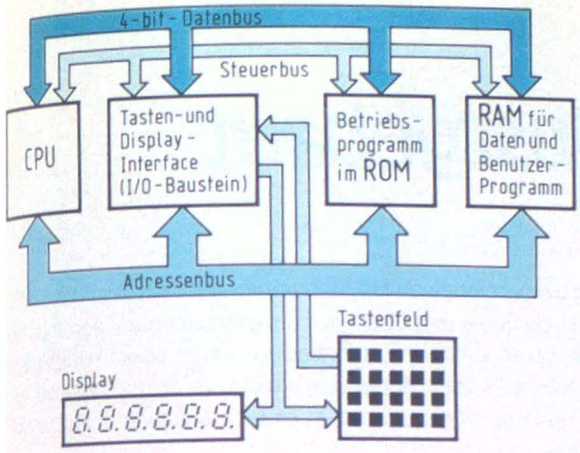


Bild 1. Blockschaltbild eines programmierbaren Taschenrechners. Im Extremfall (z. B. TI-58) können praktisch alle Funktionsblöcke auf einem einzigen Chip integriert werden

die gleiche Aufgabe benötigt, so stellt man fest, daß die Unterschiede bei einer solchen Rechnung gar nicht so groß sind. Und auch bei der Rechengenauigkeit zeigt sich ein Taschenrechner oft dem Computer überlegen.

Der Grund, warum Taschenrechner als langsam gelten, rührt allerdings nicht von der Verarbeitungszeit für höhere mathematische Funktionen her, sondern von der verhältnismäßig langen Zeit, die sie für ganz primitive Aufgaben benötigen. So braucht etwa der Rechner HP-25 rund zwei Sekunden dafür, 49 Programmschritte lang überhaupt nichts zu tun (NOP-Befehl), während ein Mikroprozessor für 49 NOP-Schritte nur etwa 150 Mikrosekunden benötigt. Dieses Mißverhältnis gilt für alle Programmoperationen, die relativ einfach sind, z. B. Vergleiche, Speicheroperationen oder Abtrennen des Dezimalteils einer Zahl. Erst bei komplexeren Operationen (sin, tan, log usw.) geht der Geschwindigkeitsvorteil der Mikrocomputer verloren.

Die geringe Geschwindigkeit der Taschenrechner bei einfachen Befehlen resultiert daraus, daß jeder Tastendruck und jeder Programmbefehl interpretativ verarbeitet wird, d. h. als Folge vieler CPU-Maschinenbefehle ausgeführt werden muß. Das Betriebsprogramm des Rechners holt einen Rechenbefehl aus dem Programmspeicher oder von der Tastatur, decodiert ihn mittels einer kleinen Software-Tabelle und führt das dem Befehl entsprechende Maschinenprogramm aus. Dann wird der nächste Rechnerbefehl geholt und decodiert, usw.

Diese Arbeitsweise ähnelt sehr den BASIC-Interpretern gängiger Mikrocomputer (PET 2001, TRS-80 usw.): Auch hier werden die vom Benutzer eingegebenen Befehle interpretativ verarbeitet. Deshalb sind auch BASIC-Computer bei bestimmten Aufgaben noch wesentlich langsamer als Computer, die direkt Maschinenprogramme ausführen. Trotzdem sind BASIC-Computer bereits deutlich schneller als Taschenrechner, nicht zuletzt deshalb, weil sie mit schnelleren Mikroprozessoren größerer Wortbreite (z. B. 8 bit oder 16 bit) arbeiten.

Genügt nicht doch ein Taschenrechner?

Viele schlecht ausgenutzt in der Gegend herumstehende Mini- und Mikrocomputer könnten ohne weiteres durch leistungsfähige programmierbare Taschenrechner ersetzt werden, wenn sie vorwiegend nur für mathematisch-arithmetische Berechnungen eingesetzt werden. Die Taschenrechner bieten oft einen größeren Rechenbereich ($10^{-99}...10^{+99}$) und eine größere Genauigkeit insbesondere bei den Logarithmen und Winkelfunktionen und stehen hier auch in der Rechengeschwindigkeit den Mikrocomputern nicht nach. Sogar die Ausgabe von Zahlen, Texten und Kurven (Plotten) ist mit Hilfe eines eingebauten oder anschließbaren Druckers oft möglich.

Die Anschaffung eines Mikrocomputers ist deshalb nur dann sinnvoll, wenn man

- sich in die Mikrocomputer-Technik einarbeiten will,
- besonderen Wert auf maschinenorientierte Steueraufgaben legt,
- das System für Textverarbeitungszwecke einsetzen will oder
- die Benutzung des Systems auch Laien durch komfortable Dialog-Programme möglich machen will.

Fazit: Auch programmierbare Taschenrechner haben nach wie vor ihre Daseinsberechtigung. Mit dem Fortschritt der Rechnertechnik werden die Grenzen zu den „echten" Mikrocomputern immer mehr zerfließen – es gibt bereits programmierbare Taschenrechner mit einem Anschluß für Kassettenrecorder zur Speicherung von Programmen und mit alphanumerischer Ein- und Ausgabe auf einem LCD-Display. (Sind das noch Taschenrechner?) Fe.

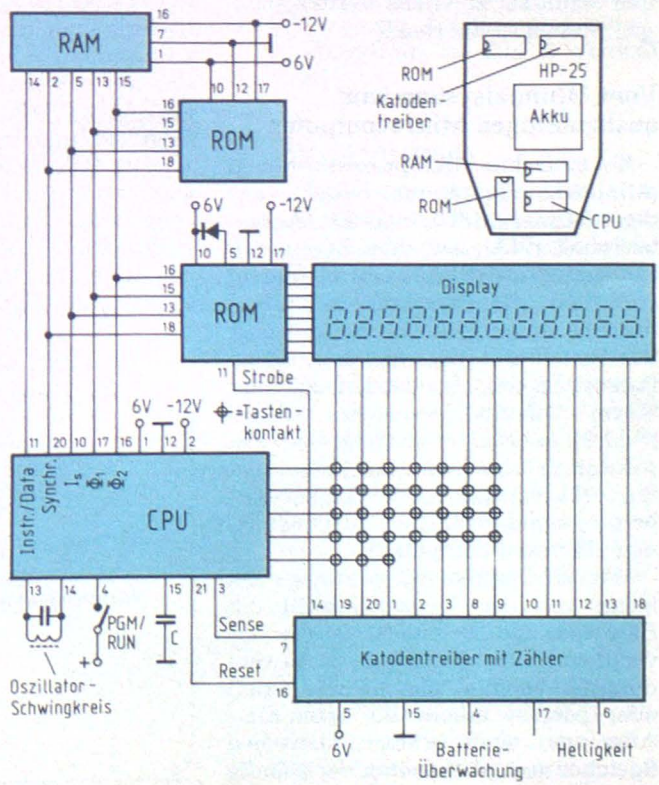


Bild 2. Innenschaltbild des Rechners HP-25. Die „CPU" ist in Wirklichkeit ein kleiner 1-Chip-Mikrocomputer, und der Datentransfer zu den übrigen Bausteinen geschieht ebenso wie die Berechnungen (um Gehäuse-Pins zu sparen) bitseriell

Ursula und Lutz May

6802-Computer – selbstgebaut

1. Teil

Die folgende Bauanleitung beschreibt einen Mikrocomputer mit dem Prozessor 6802, der auf einer Europakarte Platz findet und über ein Terminal oder ein Video-Interface mit dem Benutzer kommunizieren kann. Im Gegensatz zu vielen bisher veröffentlichten Selbstbau-Computern handelt es sich dabei also um ein Gerät, mit dem man wirklich etwas anfangen kann und das sich später auch einfach erweitern läßt. Die Software ist Motorola-kompatibel.

Eine der wichtigsten Fragen, die es beim Aufbau einer Mikrocomputer-Anlage zu lösen gilt, ist: Welcher Mikroprozessor soll verwendet werden? Dabei sind neben den technischen Daten die Verfügbarkeit und die bestehende Softwareunterstützung entscheidende Kriterien. Da hier auch der Kaufpreis eine wichtige Rolle spielt, ist die Entscheidung zugunsten des Mikroprozessors MC 6802 von Motorola gefallen. Für weniger als 400 DM ist für diesen Baustein eine komplette Programmbibliothek bestehend aus BASIC, Assembler, Text-Editor, Relocator und Disassembler erhältlich. Fertig programmierte ROMs mit der notwendigen Betriebs-Software, die von Motorola erhältlich sind, geben dem Anwender den Schlüssel zu einem leistungsfähigen System in die Hand.

Vom Minimalsystem zum ausbaufähigen Mikrocomputer

Ein einfaches Mikroprozessorsystem (Minimalkonfiguration) besteht aus dem Rechner (MPU), einer Ein/Ausgabeeinheit (PIA) und dem Programmspeicher (ROM) (*Bild 1*). Um mit diesem einfachen Mikrocomputer in Verbindung treten zu können, erweitert man das System mit einer Tastatur, einem Datensichtgerät (1) oder Drucker und einem Schreib/Lesespeicher (RAM) (*Bild 2*). Je nach verwendeter Betriebssoftware hat man nun ein Programm-Entwicklungssystem, eine Datenverarbeitungsanlage oder ganz allgemein einen „Personal Computer".

Wie komfortabel diese Anlage ist, hängt von der Ausbaufähigkeit der Hardware und besonders von der verwendeten Software ab. Je nach Anwendungsfall benötigt man mehrere seriell oder parallel arbeitende Daten-Ein-/Ausgänge, einen großen statischen Speicher zum Verarbeiten der ständig anfallenden Daten, vorbereitete Fassungen für EPROM-Bausteine der bekanntesten Fabrikate und gepufferten Daten- und Adressbus, um den Computer erweitern zu können. *Bild 3* stellt einen solchen Mikrocomputer dar, der auf einer Europakarte aufgebaut werden kann. Dem Anwender stehen mit dieser Schaltung 32 digitale Daten-Ein/Ausgänge mit 8 Steuerleitungen, zwei serielle Schnittstellen, 1 KByte RAM, 1 KByte ROM, Steckplätze für 2...4 KByte EPROM-Bausteine, ein Baudrate-Generator und gepufferte Systembusse zur Verfügung.

Systemaufbau

Der Aufbau eines Mikrocomputersystems ist nur dann sinnvoll, wenn man in der Lage ist, ein entsprechendes Betriebsprogramm (Monitorprogramm) zu erstellen, z. B. mit einem EPROM-Programmiergerät, oder die Möglichkeit hat, vom Mikroprozessorhersteller fertig programmierte ROM-Bausteine mit der Betriebssoftware zu beziehen. Die Firma Motorola vertreibt solche Programme, da sie selber Entwicklungssysteme anbietet. Eines davon, das sich für den hier beschriebenen Mikrocomputer eignet, nennt sich Minibug 3. Mit diesem Monitorprogramm kann der Anwender über eine Teletype sein Programm in Maschinensprache erstellen, es austesten und zum Laufen bringen. Mit der Betriebssystem-Erweiterungssoftware IOS können statt der Teletype eine ASCII-codierte Tastatur und ein Datensichtgerät verwendet werden, wobei damit dem Benutzer eine sehr komfortable Bildschirm-Steuerungssoftware zur Verfügung steht.

Der Aufbau dieser Programme legt die benötigten Hardwarekomponenten sowie deren Adressenzuordnung fest. Mit dem Programm Minibug 3 wird die Verbindung zu einem seriell arbeitenden System geschaffen. Dazu benötigt man ein ACIA (Asynchronous Communikations Interface Adapter), dem zur Ansteuerung der Adressbereich 8008...8009 zugeordnet wird. Zur Sicherung der variablen Daten sowie für die Stackfunktion werden 128 Byte benötigt. Um einen größeren Spielraum zu haben, ist ein 1-KByte-Speicher (RAM) mit dem Adressbereich A000...A3FF auf der Rechnerkarte untergebracht. Das Tastatur- und Datensichtgerät-Steuerprogramm IOS erfordert einen PIA (Peripherie-Inter-

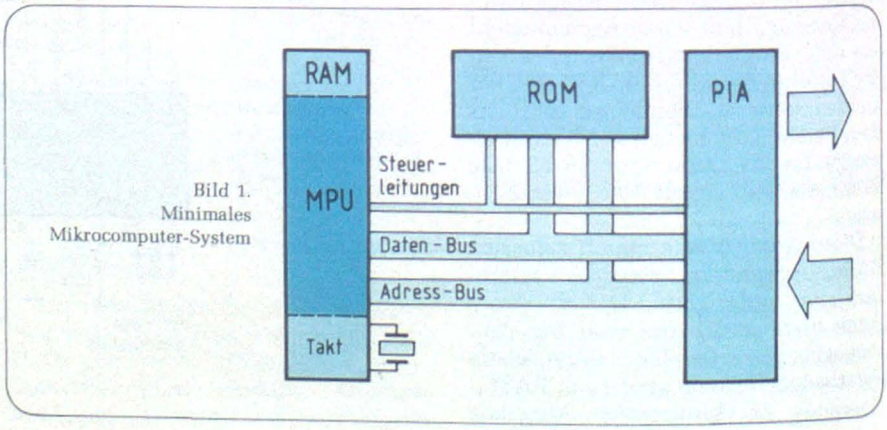


Bild 1. Minimales Mikrocomputer-System

face-Adapter), 8020...8023, um die Daten der ASCII-Tastatur aufnehmen zu können.

Da das Erweiterungsprogramm IOS keinen direkten Zugriff zum Monitorprogramm besitzt, wird die Verbindung über einen weiteren ACIA, 8010...8011, geschaffen. *Bild 4* zeigt den Signalweg zwischen den einzelnen Baugruppen. Ein vom Minibug 3 generiertes Antwortzeichen wird über den ACIA ausgegeben und über den IOS-ACIA empfangen, wodurch ein Interruptsignal ausgelöst wird. Durch den Interrupt erhält der Mikroprozessor die Anweisung, zum IOS-Programm zu springen. Soll das IOS-Programm Verbindung mit dem Monitorprogramm aufnehmen, so geschieht dies ebenfalls über die beiden ACIAs.

Um einen Drucker mit parallelem Dateneingang anschließen zu können, wird ein zweiter PIA benötigt, 8004...8007. Nach dem Restartvorgang des Rechners initiiert er, angewiesen durch das Monitorprogramm, alle Peripheriebausteine (ACIA, PIA) und überprüft deren Registerzustände. Man kann also keine Reduzierung des Hardwareaufwandes betreiben, da sonst das Monitorprogramm eine Fehlermeldung ausgibt. Beim Entwurf von eigenen Programmen kann man aber auf alle Bausteine zurückgreifen und über sie frei verfügen.

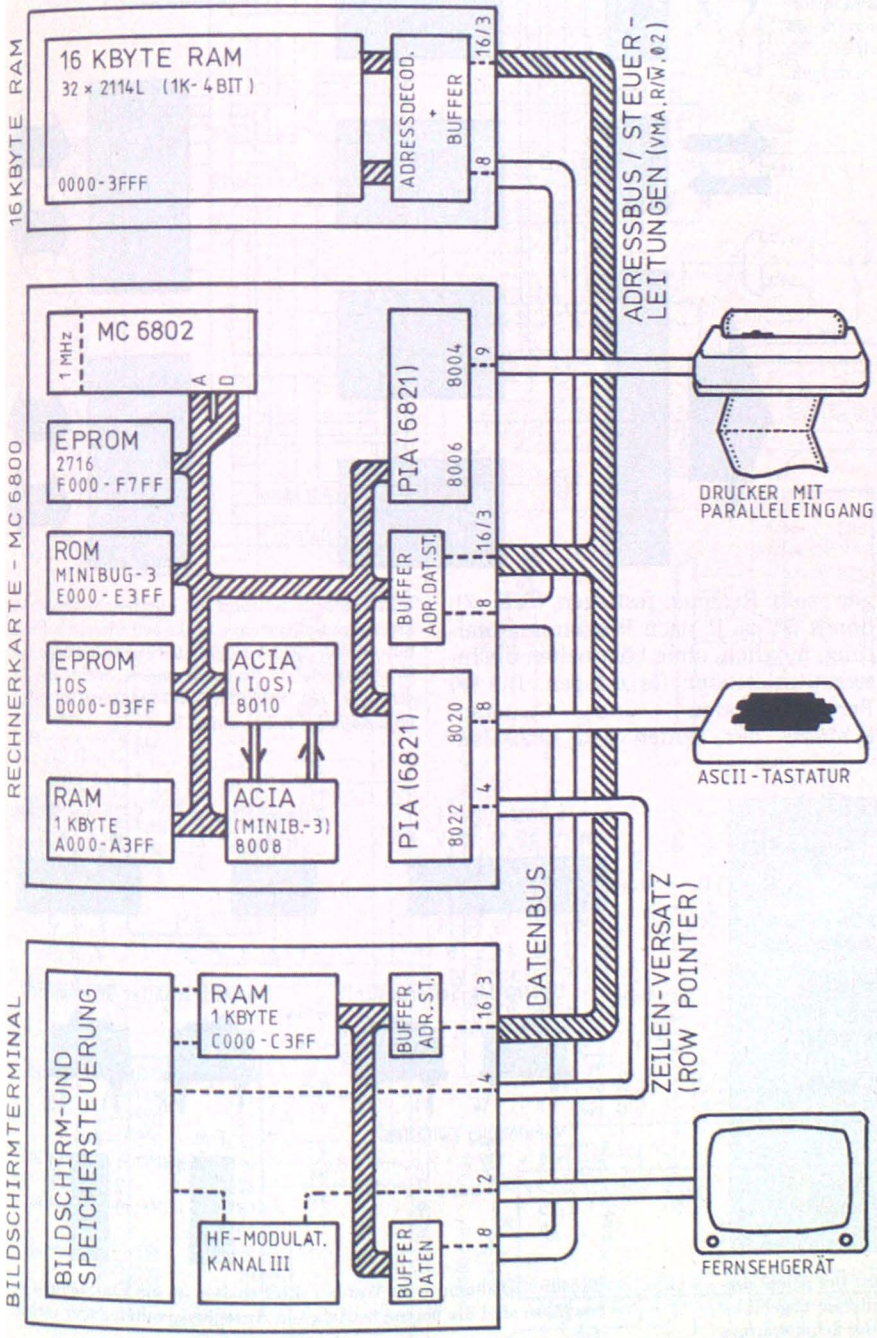


Bild 2. Erweiterter Mikrocomputer mit Ein- und Ausgabe-Einheiten

Der Mikroprozessor MC 6802

Der hier verwendete Prozessor MC 6802 ist ein monolithischer 8-bit-Mikroprozessor, der die gleiche Register- und Akkumulator-Struktur wie der MC 6800 von Motorola aufweist. Außerdem besitzt der Baustein einen Takt-Oszillator und -Treiber sowie einen abschaltbaren 128-Byte-Speicher im Adressenbereich 0000...007F. Die ersten 32 Byte, $0000...16_{16}$, haben einen eigenen Versorgungsanschluß, um die dort abgespeicherten Daten im stromsparenden Betriebszustand (low power mode), netzunabhängig durch eine Batterie, betreiben zu können. Der MC 6802 ist völlig softwarekompatibel mit dem MC 6800 und kann mit seinem 16-bit-Adressbus einen 65-KByte-Speicher verwalten. Der 8-bit-Datenbus ist bidirektional und durch das Vorhandensein von entsprechenden Steuerleitungen für den DMA-Betrieb vorbereitet.

In dieser Schaltung (*Bild 5*) wird der Rechnertakt durch einen 4-MHz-Quarz stabilisiert, was einem 1-µs-Arbeitszyklus entspricht. Soll der vom Rechner anzusteuernde Speicher, beginnend bei Adresse 0000, später erweitert werden, so ist der rechnerinterne Speicher abzuschalten. Die dazu vorgesehene Mikroprozessor-Steuerleitung IC 8, Pin 36 führt zu einem Miniaturschalter-Array, Bauelement 5, dessen genaue Aufschlüsselung in *Bild 6* verdeutlicht wird.

Restart-Steuerung

Nach dem Anlegen der Betriebsspannung braucht der Mikroprozessor zum einwandfreien Funktionsablauf 20 ms, um alle bausteininternen Systeme anlaufen zu lassen (Takt, Register- und Akkurefresh-Steuerung). Die dazu notwendige Verzögerungsschaltung besteht aus zwei NAND-Gattern (o.C.), IC 13, die 30 ms nach dem Anlegen der Betriebsspannung einen Restartimpuls an IC 8, Anschluß 40, weiterleitet. Damit auch der Anwender die Möglichkeit hat, bei Bedarf den Restartvorgang auszulösen, wird die Restart-Steuerleitung zum Busstecker, Anschluß 18 c, geführt.

Die PIA MC 6820

Zum Anschluß einer ASCII-codierten Tastatur und eines Druckers mit parallelem Dateneingang sowie zur Lösung

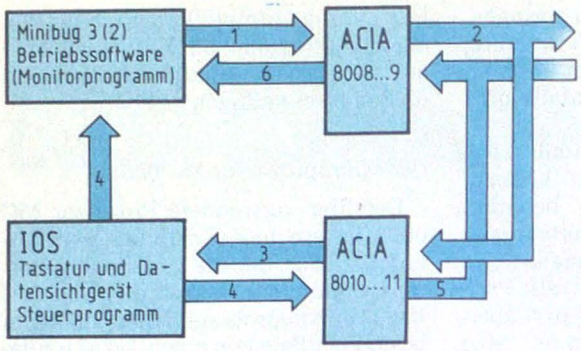


Bild 4. Datenverlauf zwischen dem Monitorprogramm und der IOS-Erweiterungssoftware: 1. Der Monitor generiert Antwortzeichen. 2. Die asynchrone Schnittstelle überträgt das Signal zum TTY oder Drucker und zur zweiten asynchronen Schnittstelle. 3. Das empfangene Signal wird durch die IOS-Software verarbeitet. 4. Vom IOS-Programm wird entweder direkt in das Monitorprogramm gesprungen oder über die asynchrone Schnittstelle ein Antwortzeichen generiert. 5. Das IOS-Antwortsignal wird in den Empfangszweig der ersten asynchronen Schnittstelle übertragen. 6. Das Monitorprogramm wertet dieses Signal aus, als wäre es von einem TTY generiert worden

anwendertechnischer Probleme werden die Peripherie-Adapter IC 6 und 2 verwendet. Diese Bausteine bilden mit zwei 8-bit-bidirektionalen Peripherie-Datenleitungen und vier Steueranschlüssen ein Interface zwischen dem Prozessor und der Computerumwelt. Für die meisten Interfaceaufgaben werden keine weiteren Logikbaugruppen benötigt.

Die Arbeitsweise der PIA wird durch den Prozessor initialisiert. Jede der 16 Datenleitungen kann durch Befehle als Datenein- bzw. -ausgang verwendet werden; ebenso können die vier Steuer/Interruptanschlüsse per Programm zur Bewältigung von einer oder mehreren Steuerfunktionen eingesetzt werden.

Mit den beiden Schalterarrays 3 und 4 lassen sich die Interruptverbindungen zum Rechner festlegen (*Bild 7*). Somit ist es, je nach Programmgestaltung, möglich, ohne Lötarbeiten die Interruptprioritäten festzulegen. Die 40 Peripherie-Daten- und Steueranschlüsse der beiden MC 6820-Bau-

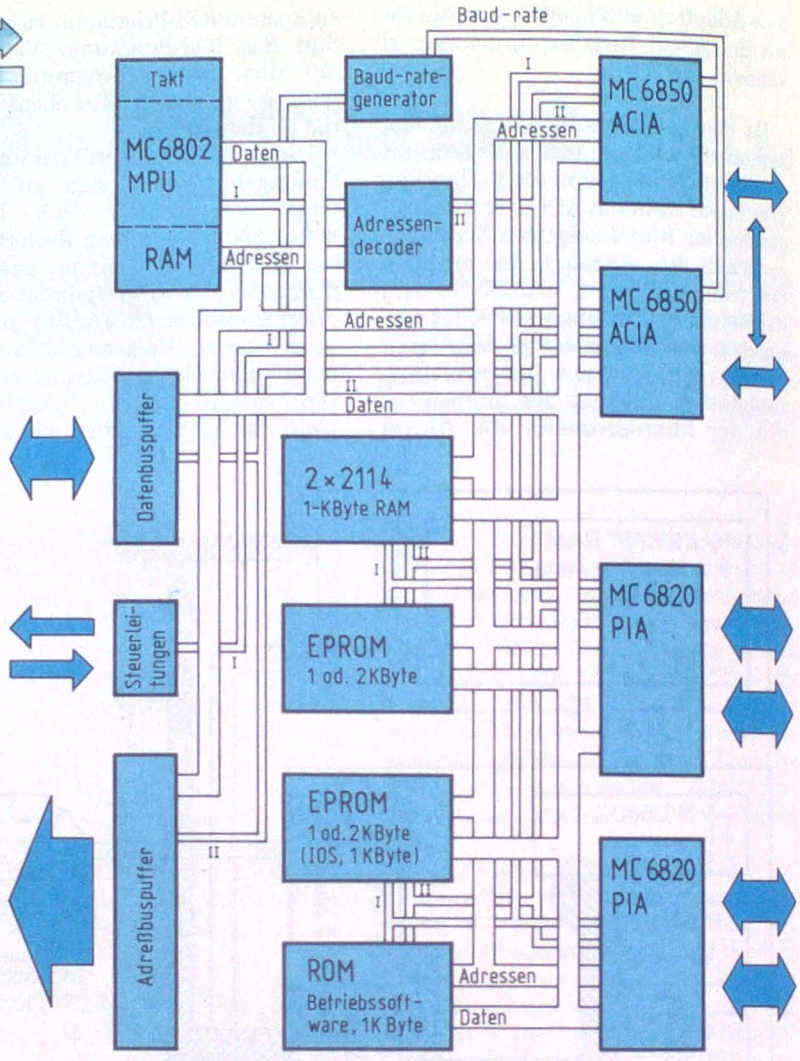


Bild 3. Blockschaltbild des hier beschriebenen Mikrocomputer-Systems. Links befinden sich die Daten-, Steuer- und Adreßbusleitungen der 64poligen VG-Steckerleiste (Reihe a und c), rechts die Anschlüsse zur Peripherie-VG-Steckerleiste (ebenfalls 64polig, Reihe a und b)

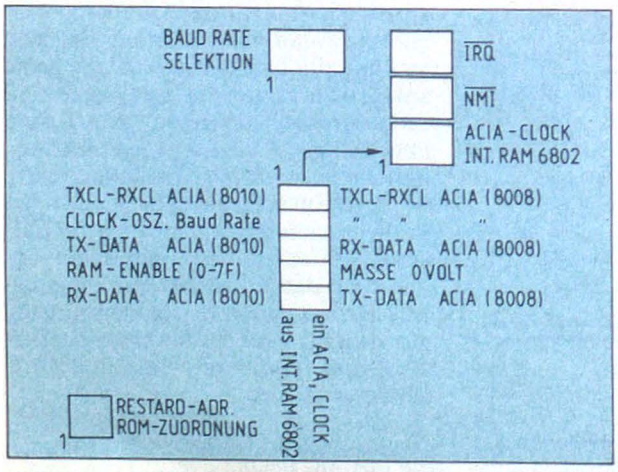


Bild 6a. Im oberen Teil ist die Mikrocomputerkarte in der Draufsicht dargestellt, wobei nur die Schalterarrays mit ihren Funktionen abgebildet sind. Der untere Bildteil gibt über den inneren Aufbau der Schalterarrays Aufschluß

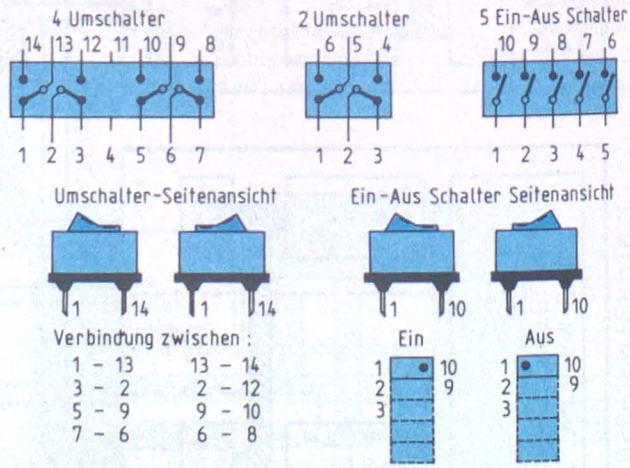


Bild 6b. Schalterarray 5. Werden die Schalter in die Ein-Stellung gebracht, so sind die beiden betriebenen Anschlüsse miteinander verbunden

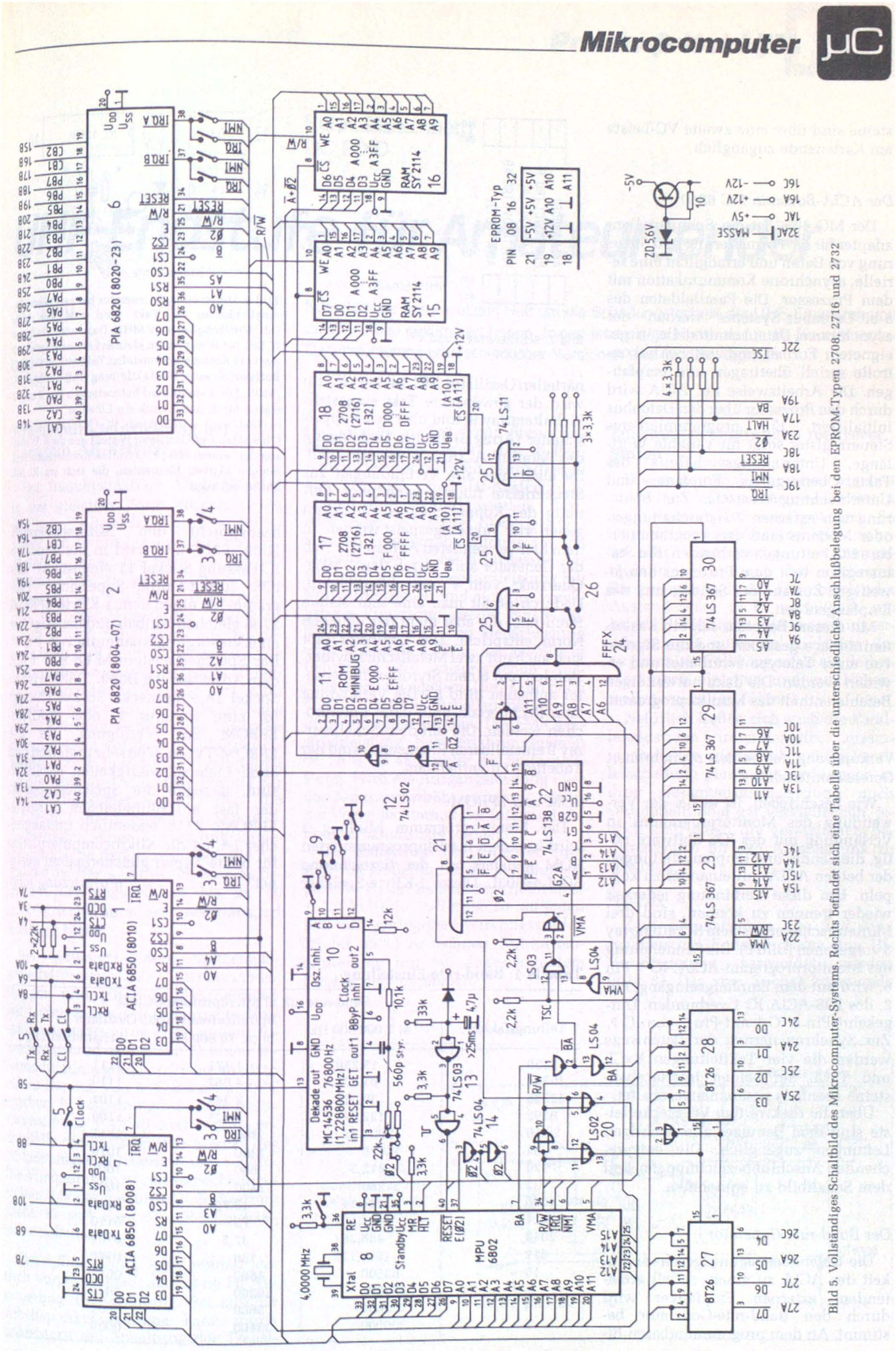


Bild 5. Vollständiges Schaltbild des Mikrocomputer-Systems. Rechts befindet sich eine Tabelle über die unterschiedliche Anschlußbelegung bei den EPROM-Typen 2708, 2716 und 2732

steine sind über eine zweite VG-Leiste am Kartenende zugänglich.

Der ACIA-Baustein MC 6850

Der MC 6850 ist ein Schnittstellenadapter für die Formatierung und Steuerung von Daten und ermöglicht eine serielle, asynchrone Kommunikation mit dem Prozessor. Die Paralleldaten des 8-bit-Datenbus-Systems werden der asynchronen Datenschnittstelle in geeignetem Format und mit Fehlerkontrolle seriell übertragen und empfangen. Die Arbeitsweise der ACIA wird durch den Prozessor über den Datenbus initialisiert. Ein programmierbares Steuerregister sorgt für variable Wortlänge, Unterteilungsverhältnis des Takts, Übertragungs-, Empfangs- und Unterbrechungskontrolle. Zur Steuerung von externen Zusatzschaltungen oder Modems sind drei programmierbare I/O-Leitungen vorhanden. Ein Statusregister teilt dem Prozessor den jeweiligen Zustand des Senders und des Empfängers mit.

Mit diesem Baustein soll ein Kassetteninterface gesteuert und die Signale von einer Teletype verarbeitet und erwidert werden. Die dazu notwendigen Befehle enthält das Monitorprogramm.

Verkopplung der beiden asynchronen Datenschnittstellen

Wie beschrieben, ist es bei der Verwendung des Monitorprogramms in Verbindung mit der IOS-Software nötig, die Sende- und Empfangsleitungen der beiden ACIAs miteinander zu koppeln. Um diese Verbindung jederzeit wieder trennen zu können, sind drei Miniaturschalter aus dem Schalterarray 5 vorgesehen (Bild 6). Die Sendeleitung des Monitorprogramm-ACIA, IC 7 Pin 6, wird mit dem Empfangseingang, Pin 2, des IOS-ACIA IC 1 verbunden. Umgekehrt Pin 6 IC 1 mit Pin 2 von IC 7. Zur Synchronisierung der Datenwege werden die vier Taktleitungen RxCL und TxCL der beiden Interfacebausteine ebenfalls zusammengeschaltet.

Über die rückwärtige VG-Steckerleiste sind dem Benutzer alle wichtigen Leitungen zugänglich. Die entsprechenden Anschlußbezeichnungen sind dem Schaltbild zu entnehmen.

Der Baud-rate-Generator

Die Datenübertragungsgeschwindigkeit des ACIA zu einem seriell arbeitenden, externen Empfänger wird durch den Baud-rate-Generator bestimmt. An dem programmierbaren Bi-

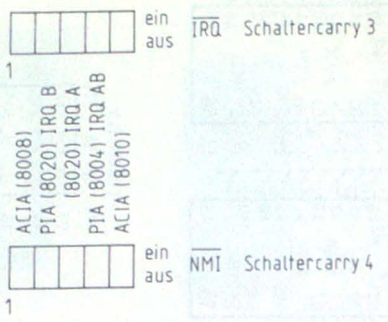


Bild 7. Schalterarray 3 und 4

närteiler/Oszillator MC 14 536 (IC 10) wird der gewünschte Takt eingestellt (Schalterarray 9) und über den Koppelschalter (Array 5) an den Takteingang des Monitor-ACIA IC7 weitergeleitet. Da auch hier alle Verbindungen zur Steckerleiste führen, kann nach Öffnung des Koppelschalters ein extern generierter Takt eingespeist werden. In dem hier verwendeten Aufbau läßt sich der Generator auch durch den 1-MHz-Quarztakt vom Prozessor ansteuern. Dadurch erhält man eine sehr stabile Baud-rate, die aber nicht der üblichen Norm entspricht. Der Oszillator läßt sich auch mit zwei Metallschichtwiderständen und einem Styroflexkondensator aufbauen (*Bild 8*). Die Verbindung zum Rechnertakt muß dann unterbrochen werden. Die mit dem Schalterarray 9 einstellbaren Frequenzen sind der *Tabelle 1* zu entnehmen.

ROM und EPROM

Das Monitorprogramm Minibug 3 wird in einem maskenprogrammierten ROM geliefert mit der Bezeichnung SCM 44503L. Dieser 1-KByte-Speicher

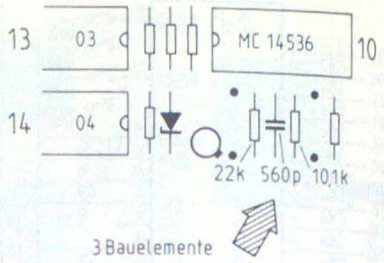


Bild 8. Der Baud-rate-Generator hat zwei Betriebsmöglichkeiten: a) Takt wird von der MPU (MC 6802) abgeleitet (1 MHz). Dazu wird Pin 3 vom IC MC 14536 mit einem kleinen Lötzinntropfen mit der vom Rechner kommenden Taktleitung (Pin 37) kurzgeschlossen. Die Taktleitung läuft direkt am IC 14536, Pin 3 vorbei und hat zwischen Pin 2 und 3 eigens für diesen Zweck ein Lötauge
b) Takt wird durch einen RC-Oszillator erzeugt (76,8 kHz). Der Oszillator besteht aus zwei Widerständen, einem 560-pF-Styroflex-Kondensator und einigen aktiven Elementen, die sich im IC MC 14536 befinden

beansprucht den Adressenbereich E000...E3FF und wird in die 24polige IC-Fassung Sockel 11 eingesteckt. Das IOS (Input-Output-Supervicer) Programm hat einen dem 1-KByte-EPROM 2708 gleichen Aufbau und benötigt nur eine Versorgungsspannung von + 5 V. Der Speicherbaustein M 68 IOS 1 mit dem Adreßbereich D000...D3FF wird in Sockel 18 eingesteckt. Sockel 17 steht für eine weiteres 1- oder 2-KByte-EPROM zur Verfügung. Da die EPROM-Typen 2708 sehr preiswert und ohne Lieferschwierigkeiten erhältlich sind, dagegen die Speicherkapazität der fast anschlußgleichen 2-KByte-EPROMs 2716 wesentlich umfangreicher ist, ist die Mikrocomputerplatine für beide Typen gleichermaßen geeignet.

(Fortsetzung folgt)

Tabelle 1: Baud-rate-Einstellung

Teilungsfaktor	Baud-rate mit a) Microprozessor-Clock a) 1 000 000 Hz	b) freilaufendem RC-Oszillator b) ca. 76 800 Hz	Schalterstellung
65536	15,258	1,171	1111
16384	61,035	4,687	1110
32768	30,517	2,343	1101
8192	122,07	9,375	1100
256	3906,25	300	1011
64	15625	1200	1010
128	7812,5	600	1001
32	31250	2400	1000
4096	244,14	18,75	0111
1024	976,562	75	0110
2048	488,28	37,5	0101
512	1953,125	150	0100
16	62500	4800	0011
4	250000	19200	0010
8	125000	9600	0001
2	500000	38400	0000

Hans Werner

Mikrocomputer

Eine kleine Einführung

28. Teil

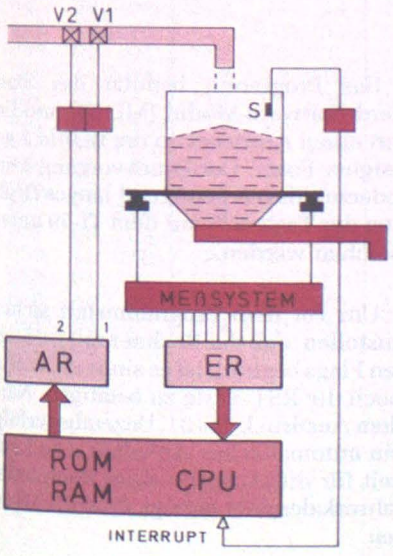
Interrupt-Anwendung

Ein typisches Beispiel ist der Stromausfall. Durch ein externes System wird ein Interruptsignal erzeugt, das Programm unterbrochen und die wichtigsten Daten in einem nicht „zerstörbaren" Speicher gesichert. Für das Mikrocomputersystem ist eine Notstromversorgung erforderlich. Weiterhin ist es sinnvoll, einen Interrupt durch Tastaturen, Anzeigen, Drucker usw. auszulösen – und zwar dann, wenn das Gerät Daten senden bzw. empfangen will.

Sollen z. B. mit einer Tastatur Werte in den Mikrocomputer gegeben werden, kann mit dem Interruptimpuls das Eingabeprogramm angesprungen werden.

Beispiel 1

Überschreitung eines Grenzwertes: Ein Behälter steht auf einer Waage. Das Wägeergebnis gelangt über ein Meßsystem in den Mikrocomputer.



Normalerweise ist Ventil V2 immer geöffnet. Wird ein bestimmter Materialfüllstand erreicht, schließt automatisch Ventil V1. Der Füllvorgang wird durch das Hauptprogramm gesteuert.

Wir wollen nicht das Hauptprogramm betrachten, sondern ein zusätzliches Sicherheitsprogramm entwikkeln.

Interruptimpuls

Überschreitet aus irgendwelchen Gründen der Materialpegel die erlaubten Grenzen (z. B.: V1 versagt), gibt Sonde S einen Impuls auf den Interrupteingang. Das Programm wird mit dem Interruptprogramm fortgesetzt.

Interruptprogramm

Es sollen die wichtigsten Daten gerettet und das Ventil V2 geschlossen werden. Wir beschränken uns auf das Wägeergebnis. Es wird in Zelle KG gespeichert. Ventil V2 ist am Ausgang 2 des Ausgaberegisters angeschlossen.
Ausgang 2 = 1: Ventil geschlossen

```
      0   SET I    1D      Interrupt initialisieren
IR ---> 52  AD    d  ....     HAUPTPROGRAMM
    240   LA AC d  512D    Daten (Gewicht) holen
    243   SP AC d   KG     und retten
    246   LA AC k   2D
    248   SP AC d  512D    Ventil V2 schließen
    251   END
```

Die Speicherplatzadressierung ist nicht fortlaufend, weil die Befehle länger als ein Byte sind. Reichen die Speicherzellen 240...255 nicht aus, kann mit einem Sprungbefehl auf andere Plätze gesprungen werden.

Beispiel 2: Wiegen und Uhrzeit

Beispiel 1 darf nicht dazu verleiten, den Interrupt als „Notsignal" anzusehen. Geschickte Programmierer kennen viele andere Einsatzmöglichkeiten. Es können z. B. mehrere Aufgaben so programmiert werden, daß sie der Mikrocomputer scheinbar gleichzeitig bearbeitet.

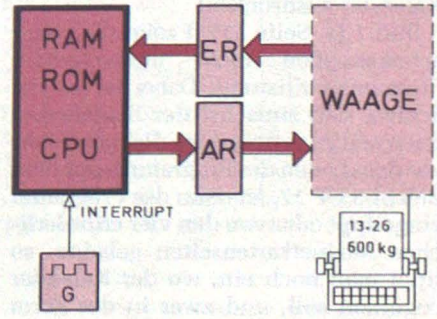
Der Prozessor ist meist viel schneller als ein peripheres Gerät. Er kann mehrere Geräte „gleichzeitig" bedienen.

Früher wurde das Programm für eine einfache Autowaage erklärt. Wir wollen an den Mikrocomputer noch einen Drucker anschließen, der das genaue Wägeergebnis auf eine Quittungskarte druckt. Damit der Beleg vollständig ist, soll auch die genaue Uhrzeit auf der Karte stehen.

Das läßt sich sehr gut mit einem Interruptprogramm machen. Uhrzeit- und Wägeprogramm wechseln sich ständig ab. Der äußere Betrachter sieht es als gleichzeitige Vorgänge. Das Uhrzeitprogramm darf allerdings nicht zu lang sein. Sonst könnte es geschehen, daß während seiner Laufzeit der Gewichtswert zu stark überschritten wird.

Ein Impulsgenerator gibt jede Minute einen Interruptimpuls, der ein kurzes Zählprogramm auslöst.

Die Uhrzeit wird in den Speicherzellen MIN und STD gespeichert und dem



Hauptprogramm zur Verfügung gestellt. An 3 Stellen kann ein Rücksprung erfolgen (POP). Da die Speicherplätze von 240...255 nicht ausreichen, ist das eigentliche Interruptprogramm ab Platz 200 untergebracht.

Vor dem ersten Impuls muß der Inhalt der Speicherzellen MIN und STD Null sein. Um Mitternacht wird der Impulsgeber angeschlossen.

BR C1 ist ein bedingter Sprungbefehl. Er wird nur dann ausgeführt, wenn eine Rechnung einen Übertrag generiert. Im Programm ist er zweimal enthalten.

Begriffe, Begriffe...

In Fachzeitschriften tauchen ab und zu die englischen Abkürzungen LIFO und FIFO auf.

LIFO:
Last in, first out (zuletzt hinein, zuerst heraus). Der LIFO wird auch Stapel- oder Kellerspeicher genannt. Das Stackregister von Punkt B60 ist z. B. ein LIFO. Die zuletzt eingespeicherten Werte werden zuerst wieder ausgelesen. Das Gegenteil davon ist der

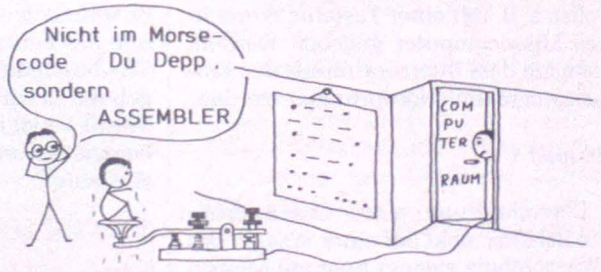
FIFO:
First in, first out (zuerst hinein, zuerst hinaus). FIFO-Speicher werden auch als Silospeicher bezeichnet. Die zuerst eingespeicherten Werte werden zuerst ausgelesen.

Ende

Wir sind am Schluß der Ausführungen. Der Streifzug durch die Technik der Mikroprozessoren ist beendet. Natürlich sind noch viele Fragen offen. Aber keine Angst, mit guten Grundlagenkenntnissen kann man sich schnell in spezielle Gebiete einarbeiten. Wer bis jetzt durchgehalten hat, darf schon auf englisch „Meikrokomputer" sagen!

```
1. Minuten   60                2. Stunden   24
           +196                            +232
            256=Übertrag                    256=Übertrag

     [1]00000000                    [1]00000000

[240]  SET C      0D   Übertrag-FF gleich 0 setzen
       JMP   d  200D   zum eigentlichen IR-Programm

200    IN    d   MIN   Anzahl der Minuten um 1 erhöhen
       AD    k  196D   Anzahl der Minuten gleich 60:
       BR C1 d  ....          POP überspringen
       POP
       LA AC k    0D   Minuten gleich Null setzen
       SP AC d   MIN

       IN    d   STD   Anzahl der Stunden um 1 erhöhen
       AD    k  232D   Anzahl der Stunden gleich 24:
       BR C1 d  ....          POP überspringen
       POP
       LA AC k    0D   Stunden gleich Null setzen
       SP AC d   STD
       POP
```

Diese Serie, die in Heft 21/1978 begann, erschien nun als Sonderheft und ist im Buch- und Zeitschriftenhandel sowie direkt beim Franzis-Verlag, Abt. Zeitschriftenvertrieb, Postfach 37 01 20, 8000 München 37, zum Preis von 9.50 DM (+ 1 DM Porto) erhältlich.



TI-59 druckt Kalender

Die alphanumerischen Druckmöglichkeiten des programmierbaren Taschenrechners TI-59 in Verbindung mit dem Thermodrucker PC 100 A erlauben es, einen vollständigen Kalender mit Überschriften für Monatsnamen und Wochentage ausdrucken zu lassen. Das geht zwar nicht allzu schnell vor sich, aber wer läßt sich schon alle Tage einen Kalender ausdrucken!

Bild 1 (s. Seite 1313) zeigt die zugegebenermaßen recht umfangreiche Programmauflistung. Dabei ist zu beachten, daß zunächst der Speicherbereich neu zu verteilen ist; dies geschieht vor dem Laden des Programms mit dem Befehl 5 OP 17. Ist dann das Programm eingetippt oder von den vier erforderlichen Magnetkartenseiten geladen, so tippt man noch ein, wo der Kalender beginnen soll, und zwar in der Form MM01.JJJJ, wobei MM der jeweilige Monat ist (1...12) und JJJJ das Jahr, das immer vierstellig eingegeben werden muß. Der Start erfolgt mit der Label-Taste A.

```
        1980.

J A N U A R

MO DI MI DO FR SA SU

   01 02 03 04 05 06
07 08 09 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

Bild 2. So sieht der Ausdruck für den Januar 1980 aus

Das Programm benutzt das Standard-Software-Modul (ML 20) und liefert einen Ausdruck in der in *Bild 2* gezeigten Form. Der Druckvorgang kann jederzeit durch genügend langes Drükken der Taste R/S auf dem TI-59 unterbrochen werden.

Um vor dem Programmstart sicherzustellen, daß der Rechner mit gelöschten Flags beginnt, ist es sinnvoll, vorher noch die RST-Taste zu betätigen. Nach dem Ausdruck des 31. Dezember erfolgt ein automatisches Anhalten. Die Laufzeit für die Ausgabe eines kompletten Jahreskalenders beträgt rund 23 Minuten.

Außer dem zum Starten benötigten Label A werden keine weiteren Labels verwendet. Dies führte zu einem Geschwindigkeitsgewinn von rund 40 %.

Ing. grad. Robert Kröpelin

000	61	GTO	063	42	STO	126	42	STO	189	02	02	264	71	SBR	339	12	12	414	08	8
001	04	04	064	00	00	127	08	08	190	57	57	265	02	02	340	43	RCL	415	42	STO
002	54	54	065	01	1	128	61	GTO	191	69	OP	266	31	31	341	10	10	416	14	14
003	61	GTO	066	42	STO	129	01	01	192	01	01	267	52	EE	342	36	PGM	417	22	INV
004	04	04	067	08	08	130	44	44	193	09	9	268	06	6	343	20	20	418	52	EE
005	65	65	068	61	GTO	131	03	3	194	42	STO	269	44	SUM	344	11	A	419	69	OP
006	61	GTO	069	01	01	132	01	1	195	11	11	270	14	14	345	01	1	420	03	03
007	01	01	070	44	44	133	42	STO	196	69	OP	271	22	INV	346	00	0	421	01	1
008	86	86	071	02	2	134	13	13	197	29	29	272	52	EE	347	00	0	422	08	8
009	61	GTO	072	07	7	135	86	STF	198	43	RCL	273	43	RCL	348	44	SUM	423	42	STO
010	03	03	073	42	STO	136	01	01	199	09	09	274	14	14	349	10	10	424	11	11
011	62	62	074	00	00	137	04	4	200	32	X:T	275	92	RTN	350	43	RCL	425	61	GTO
012	61	GTO	075	02	2	138	05	5	201	43	RCL	276	71	SBR	351	10	10	426	01	01
013	03	03	076	42	STO	139	42	STO	202	13	13	277	02	02	352	36	PGM	427	96	96
014	74	74	077	08	08	140	00	00	203	22	INV	278	20	20	353	20	20	428	71	SBR
015	61	GTO	078	61	GTO	141	03	3	204	77	GE	279	52	EE	354	12	B	429	02	02
016	04	04	079	01	01	142	42	STO	205	02	02	280	02	2	355	36	PGM	430	20	20
017	01	01	080	44	44	143	08	08	206	09	09	281	44	SUM	356	20	20	431	52	EE
018	61	GTO	081	02	2	144	73	RC*	207	83	GO*	282	14	14	357	13	C	432	04	4
019	04	04	082	09	9	145	00	00	208	11	11	283	71	SBR	358	42	STO	433	44	SUM
020	28	28	083	42	STO	146	84	OP*	209	69	OP	284	02	02	359	13	13	434	14	14
021	01	1	084	00	00	147	08	08	210	05	05	285	31	31	360	83	GO*	435	71	SBR
022	07	7	085	02	2	148	69	OP	211	98	ADV	286	44	SUM	361	12	12	436	02	02
023	42	STO	086	42	STO	149	30	30	212	98	ADV	287	14	14	362	71	SBR	437	31	31
024	00	00	087	08	08	150	97	DSZ	213	87	IFF	288	22	INV	363	02	02	438	52	EE
025	03	3	088	61	GTO	151	08	08	214	01	01	289	52	EE	364	76	76	439	02	2
026	42	STO	089	01	01	152	01	01	215	04	04	290	43	RCL	365	69	OP	440	44	SUM
027	08	08	090	44	44	153	44	44	216	86	86	291	14	14	366	01	01	441	14	14
028	61	GTO	091	03	3	154	69	OP	217	61	GTO	292	92	RTN	367	01	1	442	43	RCL
029	01	01	092	02	2	155	05	05	218	03	03	293	76	LBL	368	02	2	443	14	14
030	44	44	093	42	STO	156	98	ADV	219	15	15	294	11	A	369	42	STO	444	22	INV
031	02	2	094	00	00	157	87	IFF	220	43	RCL	295	42	STO	370	11	11	445	52	EE
032	00	0	095	03	3	158	00	00	221	09	09	296	10	10	371	61	GTO	446	69	OP
033	42	STO	096	42	STO	159	01	01	222	55	÷	297	22	INV	372	01	01	447	03	03
034	00	00	097	08	08	160	73	73	223	01	1	298	59	INT	373	96	96	448	00	0
035	03	3	098	61	GTO	161	04	4	224	00	0	299	52	EE	374	71	SBR	449	42	STO
036	42	STO	099	01	01	162	09	9	225	95	=	300	04	4	375	02	02	450	11	11
037	08	08	100	44	44	163	42	STO	226	59	INT	301	22	INV	376	20	20	451	61	GTO
038	61	GTO	101	03	3	164	00	00	227	85	+	302	52	EE	377	52	EE	452	01	01
039	01	01	102	06	6	165	04	4	228	01	1	303	99	PRT	378	06	6	453	96	96
040	44	44	103	42	STO	166	42	STO	229	95	=	304	98	ADV	379	42	STO	454	71	SBR
041	02	2	104	00	00	167	08	08	230	92	RTN	305	43	RCL	380	14	14	455	02	02
042	02	2	105	04	4	168	86	STF	231	09	9	306	10	10	381	71	SBR	456	57	57
043	42	STO	106	42	STO	169	00	00	232	32	X:T	307	36	PGM	382	02	02	457	69	OP
044	00	00	107	08	08	170	61	GTO	233	43	RCL	308	20	20	383	31	31	458	04	04
045	02	2	108	61	GTO	171	01	01	234	09	09	309	14	D	384	52	EE	459	03	3
046	42	STO	109	01	01	172	44	44	235	55	÷	310	65	×	385	04	4	460	42	STO
047	08	08	110	44	44	173	22	INV	236	01	1	311	03	3	386	44	SUM	461	11	11
048	61	GTO	111	03	3	174	86	STF	237	00	0	312	95	=	387	14	14	462	61	GTO
049	01	01	112	09	9	175	00	00	238	95	=	313	42	STO	388	22	INV	463	01	01
050	44	44	113	42	STO	176	00	0	239	22	INV	314	11	11	389	52	EE	464	96	96
051	02	2	114	00	00	177	42	STO	240	59	INT	315	69	OP	390	43	RCL	465	71	SBR
052	04	4	115	03	3	178	14	14	241	65	×	316	00	00	391	14	14	466	02	02
053	42	STO	116	42	STO	179	69	OP	242	01	1	317	01	1	392	69	OP	467	76	76
054	00	00	117	08	08	180	00	00	243	00	0	318	02	2	393	02	02	468	69	OP
055	02	2	118	61	GTO	181	01	1	244	95	=	319	32	X:T	394	01	1	469	04	04
056	42	STO	119	01	01	182	42	STO	245	67	EQ	320	43	RCL	395	05	5	470	06	6
057	08	08	120	44	44	183	09	09	246	02	02	321	10	10	396	42	STO	471	42	STO
058	61	GTO	121	04	4	184	83	GO*	247	52	52	322	55	÷	397	11	11	472	11	11
059	01	01	122	02	2	185	11	11	248	85	+	323	01	1	398	61	GTO	473	43	RCL
060	44	44	123	42	STO	186	69	OP	249	01	1	324	00	0	399	01	01	474	09	09
061	02	2	124	00	00	187	00	00	250	95	=	325	00	0	400	96	96	475	32	X:T
062	05	5	125	03	3	188	71	SBR	251	92	RTN	326	95	=	401	71	SBR	476	43	RCL
									252	01	1	327	59	INT	402	02	02	477	13	13
									253	02	2	328	67	EQ	403	20	20	478	67	EQ
									254	22	INV	329	01	01	404	44	SUM	479	01	01
									255	52	EE	330	31	31	405	14	14	480	96	96
									256	92	RTN	331	65	×	406	43	RCL	481	69	OP
									257	71	SBR	332	01	1	407	14	14	482	05	05
									258	02	02	333	00	0	408	69	OP	483	61	GTO
									259	20	20	334	85	+	409	02	02	484	01	01
									260	52	EE	335	01	1	410	71	SBR	485	96	96
									261	08	8	336	01	1	411	02	02	486	98	ADV
									262	42	STO	337	95	=	412	31	31	487	98	ADV
									263	14	14	338	42	STO	413	52	EE	488	92	RTN

Bild 1. Das vollständige TI-59-Programm zum Ausdrucken eines Kalenders

Ursula und Lutz May

6802-Computer – selbstgebaut

2. Teil

Im zweiten Teil dieser Bauanleitung wird zunächst die Beschreibung des Systemaufbaus abgeschlossen. Anschließend werden Hinweise für den Zusammenbau der Computer-Platine, zur Inbetriebnahme sowie zur Systemerweiterung gegeben.

In die 24poligen EPROM-Stecksockel 17 und 18 können die Festwertspeichertypen 2708 und 2716 eingesetzt werden. Da die beiden Speicherversionen unterschiedliche Versorgungs- und Steueradressenanschlüsse haben, muß vor dem Einsetzen des jeweiligen EPROMs eine entsprechende Drahtbrücke gezogen werden. Dazu befindet sich unter dem Quarz ein Programmierfeld, dessen Leiterbahnführung und -bezeichnung in *Bild 9* dargestellt ist. *Bild 10* zeigt einige Programmierbeispiele in Abhängigkeit von den verwendeten ROM-Bausteinen.

Der Restartvorgang löst beim Mikroprozessor eine Startroutine aus, bei der er die Startadresse des abzuarbeitenden Programms erfragt. Dies geschieht durch Auslesen der beiden letzten Speicheradressen (FFFE...F_{16}). Mit dem Schalterarray 26 kann dieser Adressenwert verändert werden, so daß beim Restartvorgang entweder die Adressen FFFE...F_{16}, E7FE...F_{16} oder D7FE...F_{16} (bzw. D3FE...F_{16}) abgefragt werden (*Bild 11*).

Bei den älteren 1-KByte-EPROM-Bausteinen, Typ 2708, werden noch 3 Versorgungsspannungen benötigt: + 12 V, + 5 V und – 5 V. Da – 5 V ein unüblicher Spannungswert ist, befindet sich auf der Platine ein – 5-V-Spannungsregler, der aus einer Z-Diode, einem Widerstand und einem Transistor besteht. Es können nun Spannungswerte bis – 25 V über die vordere VG-Steckerleiste, Anschluß 16c, eingespeist werden.

Statischer Speicher (RAM)

Das Monitorprogramm benötigt zur Inerruptabarbeitung ACIA- und PIA-Steuerung sowie zur Bewältigung der Stackoperationen 128 Byte Speicherplätze. Auch braucht der Anwender einen Speicher, in dem er seine Programme oder die zu verwaltenden Daten ablegen kann. Aus Platzgründen hat sich die Verwendung von zwei 1-K×4-bit-Speicherelementen, IC 15 und 16, angeboten, die im Adreßbereich A 000...A3FF vom Rechner angesteuert werden können. Dem Anwender stehen außer den ersten 128 Byte, A 000 bis A07F, alle Speicherplätze zur freien Verfügung. In *Tabelle 2* wird ein Überblick über die Speicherverwaltung der ersten 125-Byte durch das Monitorprogramm gegeben.

Adressendecodierung

Alle bisher besprochenen Baugruppen, die vom Mikroprozessor angesprochen werden sollen, müssen für den Rechner eindeutig adressierbar sein. Gibt er zum Auslesen eines Registers eine entsprechende Adresse aus, dann erwartet der Prozessor, daß sich nur eine 8-bit-Datenzelle meldet und nicht mehrere. Dazu bedarf es einer Logik, die um so aufwendiger wird, je genauer der Adreßbereich einer Baugruppe definiert werden soll. Auch hier galt es, Kompromisse zu schließen, denn der zur Verfügung stehende Platz auf einer Europakarte ist begrenzt. So wurde der gesamte Bereich von Adresse 8000...8FFF$_{16}$ zur Verwaltung der vier Interface-Bausteine (ACIA, PIA) verwendet. *Tabelle 3* zeigt die Adressenverteilung auf der Rechnerkarte und die nicht verwendeten, zur Erweiterung des Mikrocomputers zur Verfügung stehenden Adreßblöcke.

Mit dem Decoder-Baustein 74LS138, IC 22, werden die acht höherwertigen 4-K-Adreßblöcke decodiert (8000,

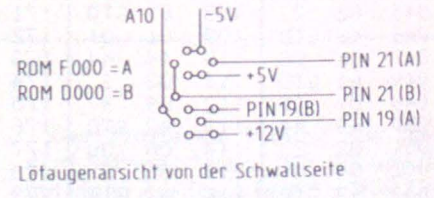


Bild 9. Adressen- und Versorgungseinstellung für die ROM- bzw. EPROM-Plätze F000...FFFF und D000...DFFF. Unter dem Quarz befinden sich auf der Karte, von der Schwallseite zugänglich, 14 Lötaugen, die in Abhängigkeit ein von den verwendeten ROMs miteinander verbunden werden müssen

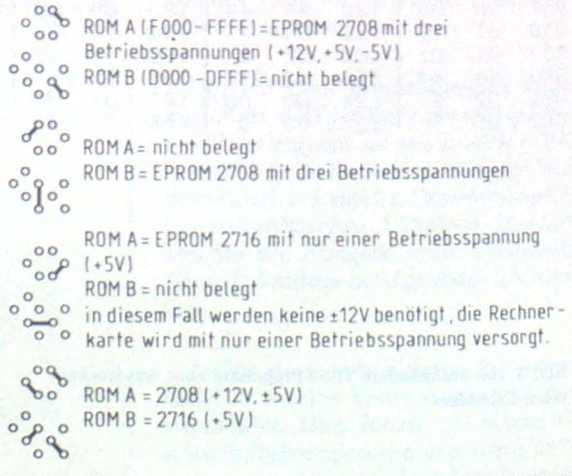


Bild 10. Hardware-Programmierbeispiele für verschiedene EPROM-Bausteine

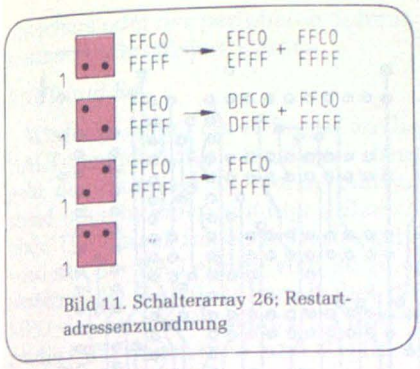


Bild 11. Schalterarray 26; Restartadressenzuordnung

9000...F000). Über Anschluß 4, IC 22, teilt der Prozessor dem Decodierer über die VMA-Leitung mit, ob die momentan ausgegebene Adresse gültig ist oder nicht. Mit dem 8fach NAND-Gatter, IC 24, werden die Adressen FFF8...FFFF decodiert und über die AND-Gatter, IC 25, mit den ROM-Steueradressen gekoppelt. Durch das Schalterarray 26 kann der Anwender bestimmen, welcher der drei auf der Karte befindlichen ROM-Bausteine auf die Datenbefragung des Rechners bei einem Restart- oder Interruptsignal reagieren soll. Für die Ansteuerung des 1-KByte-RAM wird das Taktsignal Φ2 mit der decodierten Adresse (A000...A3FF) verknüpft. Dies geschieht mit dem NOR-Gatter 74LS02, IC 12, Pin 11...13. Durch diese Synchronisierung soll eine eindeutige Datenübernahme in den Speicher gewährleistet werden.

Zur Ansteuerung der Peripherie-Interfacebausteine (ACIA, PIA) wird das decodierte Adreßsignal (8000...8FFF) von Anschluß 15, IC 22, verwendet. Die exakte Definition der einzelnen Bausteine wird mit den Chip-Select-Eingängen am jeweiligen Interface realisiert. So setzt sich die Ansprechadresse der PIA (8020...23) aus den Adreßleitungen A0, A1 und A5 sowie der decodierten Basisadresse 8000 zusammen. Eine genauere Bestimmung der Ansprechadresse ist nicht nötig, da im gesamten Bereich der Adresse 8000...8FFF$_{16}$ nur die vier Interface-Bausteine liegen.

Tabelle 2: Arbeitsspeicherverwaltung durch das Betriebsprogramm Minibug 3

A000, A001	IRQ Startadresse
A006, A007	NMI Startadresse
A00C	Echo Flag; sende alle empfangenen Daten als Bestätigung zurück
A032, A033	Startadresse des Stackpointers (wird vom Monitor auf A07F festgelegt)
A034, A035	Adressenspeicher für SWI-Funktion (I)
A036, A037	Adressenspeicher für SWI-Funktion (II)

Der Anwender kann seinem Programm entsprechend die Werte in den hier angegebenen Speicherplätzen verändern.

Tabelle 3: Adressenverteilung auf der Rechnerkarte

0000...007F	RAM, abschaltbar
8000...8FFF	Peripherie-Elemente
A000...AFFF	RAM, Funktionsbereich von A000...A3FF
D000...DFFF	ROM, EPROM, Funktionsbereich D000...D3FF bzw. D7FF
E000...EFFF	ROM, Funktionsbereich E000...E3FF
F000...FFFF	EPROM, Funktionsbereich F000...F3FF bzw. F7FF

Alle hier nicht aufgeführten Adressen kann der Anwender zur Computererweiterung verwenden.

Steuerung der Daten- und Adreßbus-Puffer

Um den Mikrocomputer erweitern zu können, ist es notwendig, dem Anwender Zugriff zum Daten- und Adreßbus sowie den Steuerleitungen zu ermöglichen. Da die Zusatzsysteme nicht wahllos das Bussystem ansteuern dürfen – sie würden sonst den Rechnerablauf empfindlich stören – müssen entsprechende Schutzschaltungen vorgesehen werden.

Der Anschluß zum Datenbus geschieht über die beiden 4fach bidirektionalen Buserweiterungsbausteine MC 6880, IC 27 und 28. Die jeweilige Datentransportrichtung wird vom Prozessor über den R/$\overline{W}$-Anschluß (lesen/schreiben), IC 8, Pin 34, bestimmt. Gilt die vom Rechner ausgegebene Adresse einem auf der Karte befindlichen Baustein, dann generiert das 8fach-NAND-Gatter IC 21 ein Sperrsignal, das zusammen mit dem R/$\overline{W}$-Signal die Datenbus-Erweiterungsbausteine steuert. Beim Adreßbus liegt eine derartige Steuerung nicht vor. Hier werden alle vom Rechner ausgelesenen Adressen übertragen. Lediglich für den DMA-Betrieb lassen sich die drei Pufferbausteine 74LS367, IC 23, 29 und 30, in den Three-state-Zustand versetzen.

Außer diesen beiden Bussystemen werden noch Steuerleitungen über die 64polige VG-Steckerleiste herausgeführt: VMA, R/$\overline{W}$, IRQ, NMI, Reset, Φ2, HALT, BA und TSC.

VMA

Diese Signalleitung gibt dem Prozessor bekannt, ob auf dem Adreßbus eine gültige Adresse ausgegeben wird oder nicht. Sie wird zur Freigabe von Speicherschaltungen und der peripheren Schnittstellen wie PIA und ACIA verwendet. Die Adresse ist gültig, wenn VMA=1 ist.

R/$\overline{W}$

Mit dieser Lese/Schreib-Leitung erfahren die peripheren Einheiten, ob sich die MPU in einem Lese- (1) oder Schreibzustand (0) befindet. Wird der Rechner angehalten, dann geht die Signalleitung in den hochohmigen Zustand über (three-state), ebenso, wenn die TSC-Leitung mit 1 angesteuert wird.

$\overline{IRQ}$

Bringt ein Peripherie-Baustein diese Leitung in den Null-Zustand, so unterbricht der Mikroprozessor die Abarbeitung des derzeitig laufenden Programms und holt sich von den Speicherplätzen FFF8 und FFF9 die Startadresse des als nächstes zu bearbeitenden Programms. Ist jedoch im Bedingungsregister der MPU das Unterbrechungsmaskenbit gesetzt, dann wird das $\overline{IRQ}$-Signal so lange ignoriert, bis das Bit gelöscht wird. Die Unterbrechung der Programmabarbeitung erfolgt erst nach vollständiger Durchführung der letzten Maschinenbefehlsanweisung. Danach werden die Inhalte der MPU-Register (Befehlszähler, Indexregister, Akkumulator A und B sowie das Bedingungsregister) in den Stack abgelegt.

$\overline{NMI}$

Es handelt sich hierbei um eine zweite Unterbrechungssteuerleitung, die in der Funktion der der $\overline{IRQ}$-Leitung entspricht. Der einzige Unterschied besteht darin, daß der Zustand des Unterbrechungsmaskenbits im Bedingungsregister nicht beachtet wird.

$\overline{Reset}$

Dieser Rechnersteuereingang wird zum Starten der MPU beim Einschalten oder nach einem Stromausfall benötigt. Ebenso kann damit zu jedem Zeitpunkt nach dem Einschalten das System erneut initiiert werden. Beim Übergang von 0 auf 1 wird der Inhalt der beiden Speicherplätze FFFE und FFFF in den Befehlszähler geladen. Dieser 16-bit-Wert gilt dann als Startadresse für das nun abzuarbeitende Programm. Gleichzeitig wird das Unterbrechungsmaskenbit gesetzt.

Φ2

Der Arbeitstakt des Rechners wird über die Steuerleitung Φ2 übertragen. Damit können weitere Baugruppen des

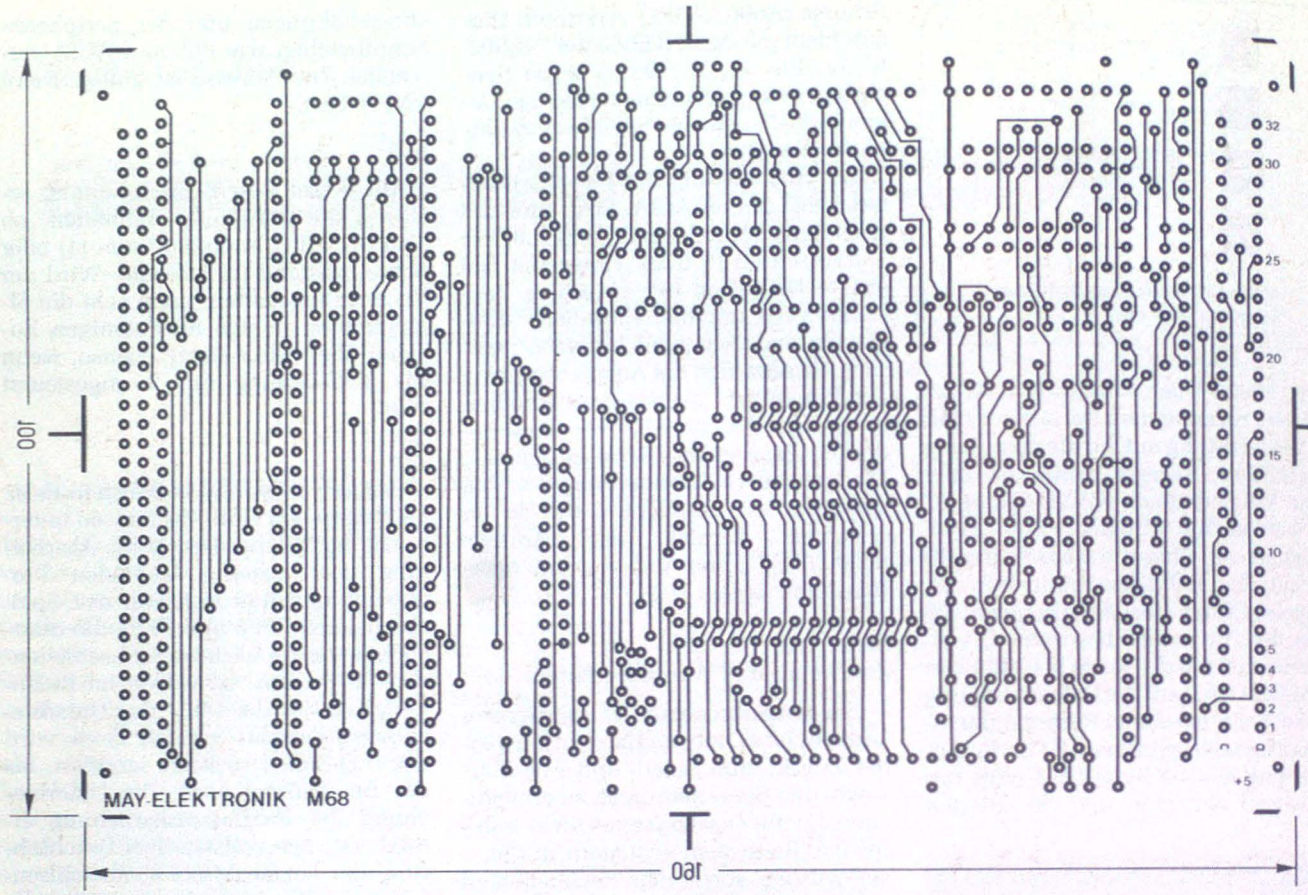


Bild 12. Die Lötseite der Computer-Europakarte

Bild 13. Bauteileseite der doppelseitigen, durchkontaktierten Platine. Fertig ist die Karte von den Verfassern erhältlich (Schlosserbreite 6, 8018 Grafing)

Speichers oder der peripheren Schnittstellen synchronisiert werden.

HALT und BA

Wird von außen ein H-Signal an die HALT-Steuerleitung angelegt, dann geht der Prozessor in einen Haltezustand oder in einen untätigen Betrieb über. Die darauf folgende Busfreigabe wird mit der BA-Leitung vom Rechner bestätigt. Ist BA=0, so verarbeitet die MPU das Steuerprogramm, bei BA=1 ist die MPU angehalten, und alle internen Aktivitäten sind gestoppt. Die Ausgänge der Daten- und Adreßbus-Puffer gehen daraufhin in den hochohmigen Zustand (three-state) über.

TSC

Hiermit kann der Adreßbus des Rechners kurzzeitig in den hochohmigen Zustand versetzt werden. Gleichzeitig wird der Rechnertakt angehalten. Es können nun für maximal 700 ns direkte Datenzugriffe in jeden beliebigen Speicher vorgenommen werden. Da der Mikroprozessor nur über dynamische Register verfügt, würden bei längerer Unterbrechung die prozessorinternen Daten verfallen.

Zusammenbau der Computer-Platine

Durch den kompakten Schaltungsaufbau ist die Leiterbahnführung sehr gedrängt. Daher ist es bei solchen Leiterplatten notwendig, vor dem Zusammenbau die Qualität der einzelnen Verbindungszüge auf Unterbrechungen oder gegebenenfalls auf Kurzschlüsse zu untersuchen. Unterläßt man eine derartige Überprüfung, muß man mit zum Teil langen Fehlersuchzeiten auf der gelöteten Platine rechnen. Das gleiche gilt auch für das Löten. Sauberes Arbeiten ist oberstes Gebot. Versehentlich entstandene Lötzinn-Kurzschlußbrücken müssen sofort mit Entlötlitze (nicht mit Lötzinnsaugpumpe) entfernt werden. *Bild 12* zeigt die Lötseite, *Bild 13* die Bauteilseite und *Bild 14* die Bestückung.

Beim Aufbau einer solch umfangreichen Schaltung, die zudem aus hochintegrierten Bausteinen besteht, ist es sinnvoll, für alle Schaltkreise IC-Sockel zu verwenden. Ein durch unsachgemäße Handhabung zerstörter Baustein läßt sich so viel leichter auswechseln.

Die folgende Montagefolge hat sich in unserem Fall als günstig erwiesen: Als erstes die IC-Sockel und Miniaturschalter, danach die Widerstände, Dioden, Kondensatoren und den Transistor einbauen; zuletzt kommen die Steckerleisten, die Tantalperlen und der Quarz.

Die integrierten Schaltkreise werden in einer entladenen Umgebung nach den Angaben des Bestückungsplans in die entsprechende IC-Fassung eingesetzt. Entladene Umgebung heißt, daß man die Bausteine vor statischen Überspannungen, wie sie durch Reiben an Nylon-Bekleidung oder durch Laufen über Kunststoff-Teppichböden entstehen, schützen muß, da außer den TTL-Schaltungen die Bausteine darauf sehr empfindlich reagieren. Zum Schluß ist jeder einzelne Anschluß zu überprüfen, ob er auch richtig verlötet wurde.

Hat man der Mikrocomputerkarte schon eine Verwendung zugedacht, in der eine Erweiterung des Systems nicht in Frage kommt, dann entfallen die Pufferbausteine IC 27, 28, 23, 29 und 30 sowie deren Ansteuerlogik-Bausteine IC 20 und 21. Um die Miniaturschalter einzusparen, können auch an deren Stelle Drahtbrücken eingesetzt werden. In diesem Fall kann das 3fach-AND-Gatter IC 25 ebenfalls durch Drahtbrükken ersetzt werden.

Inbetriebnahme der Karte

Mit einem Ohmmeter wird überprüft, ob zwischen den Versorgungsanschlüssen ein Kurzschluß besteht, wobei ein Widerstandswert von ca. 10 Ohm zwischen dem Masseanschluß und dem +5-V-Anschluß erwartet werden kann. Bei dieser Kontrolle sollen auch die +12-V-(Anschluß 16a) und –12-V-Kontakte (Anschluß 16c) mit einbezogen werden.

Wenn die Mikrocomputerkarte an das Netzteil (vorerst nur + 5 V) angeschlossen wurde, kann ein grundle-

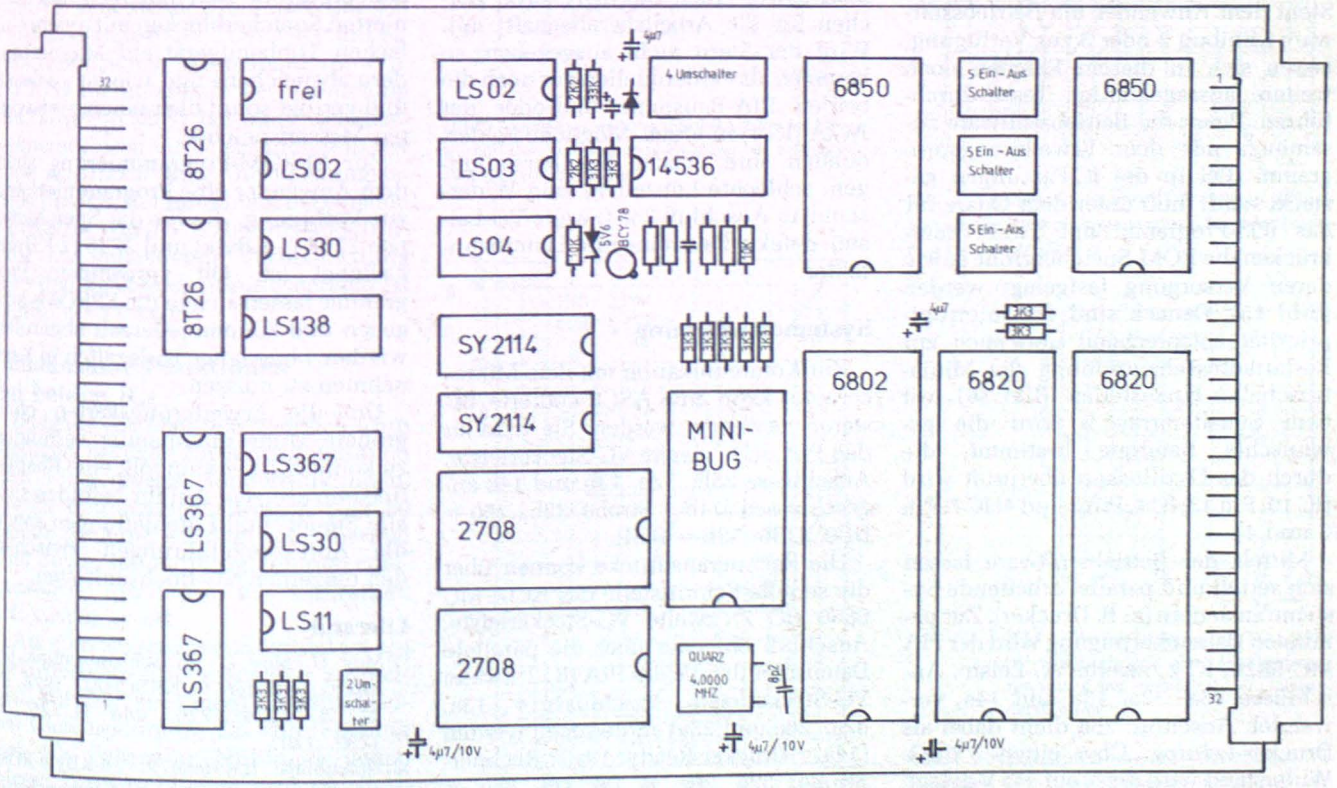


Bild 14. Bestückungsplan zu den Bildern 12 und 13. Es empfiehlt sich unbedingt die Verwendung eines Lötkolbens mit feiner Spitze

gender Fehler durch Überhitzen eines Schaltkreises in der ersten Minute festgestellt werden. Der 1-KByte-Speicher sowie der Mikroprozessor selbst nehmen nach 5...10 Minuten beachtliche Temperaturen an, was an deren hoher Verlustleistung liegt. In einem abgeschlossenen Aufbau sollte man daher auf eine Lüftungsmöglichkeit achten (größere Freiräume, Lüftungsschlitze). Mit dem Oszilloskop wird an den Anschlüssen 39 bzw. 37 des Mikroprozessorbausteins MC 6802 die Arbeit des rechnereigenen Taktgenerators nachgeprüft: Anschluß 39 = 4 MHz, Anschluß 37 = 1 MHz. Ebenfalls mit dem Oszilloskop werden die Aktivitäten aller Daten- und Adreßbusleitungen kontrolliert, es müssen auf allen 24 Leitungen (Mikroprozessoranschlüsse 9...20 und 22...33) ständig wechselnde Signale festzustellen sein.

Durch Anlegen eines 0-Signals an den Resetanschluß (VG-Leiste, Kontakt 18c) müssen alle Aktivitäten auf den Daten- und Adreßbusleitungen aufhören und nach dem Freigeben der Resetleitung wieder von neuem beginnen. Sind die bisherigen Messungen erfolgreich abgeschlossen worden, dann kann man davon ausgehen, daß die wichtigsten Verbindungen von und zum Prozessor funktionsfähig sind (nicht einbezogen sind die Steuerleitungen).

Ohne geeignete Meßmittel wird eine weitere Untersuchung schwieriger. Steht dem Anwender die Betriebssoftware Minibug 2 oder 3 zur Verfügung, lassen sich in diesem Rahmen noch weitere aussagekräftige Tests durchführen. Bevor die Betriebssoftware zusammen mit dem Erweiterungsprogramm IOS in die IC-Fassungen gesteckt wird, muß unter dem Quarz für das IOS-Programm mit zwei Drahtbrücken die ROM-Speichergröße sowie deren Versorgung festgelegt werden (*Bild 15*). Danach sind der Interruptpriorität entsprechend und auch zur Restartadressenzuordnung die Miniaturschalter einzustellen (*Bild 16*). Mit dem Schalterarray 9 wird die gewünschte Baudrate bestimmt, die durch das Oszilloskop überprüft wird (IC 10, Pin 13; IC 1, Pin 3 und 4; IC 7, Pin 3 und 4).

Mittels der Betriebssoftware lassen sich seriell und parallel arbeitende Systeme ansteuern (z. B. Drucker). Zur parallelen Datenübertragung wird der PIA MC 6820, IC 2, zweite VG-Leiste, Anschlüsse 25a...32a, 13a und 14a, verwendet. Anschluß 25a dient dabei als Drucker-Leitung. Über einen 3,3-kΩ-Widerstand wird diese auf + 5 V gelegt. Liegt dieser Anschluß auf Masse (0), dann nimmt das Rechnersteuerprogramm an, daß ein Drucker angeschlossen ist. Daraufhin unterbricht der Mikrocomputer seine Arbeit, bis über den Anschluß 14a ein Handshake-Signal empfangen wird oder 25a wieder den Wert 1 annimmt.

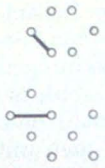


Bild 16. Schaltereinstellung auf der Mikrocomputerkarte bei Verwendung der Minibug-3- und IOS-Software

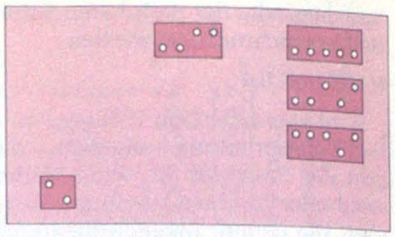


Bild 15. Plazierung der Drahtbrücken zur Definition des verwendeten ROM-Bausteins (IOS-Erweiterungssoftware)

Nach dem Betätigen der Restartfunktion muß der IOS-ROM (IC 18, Pin 20) vom Rechner angesteuert werden. Auf dem Oszilloskop müssen mehrere 0-Impulse (1 bis mehrere ms breit) erkennbar sein. Am seriellen Ausgang des ACIA MC 6850 (IC 7, Pin 6) werden im 16fach gedehnten Rhythmus der gewählten Baudrate einige TTL-Signale generiert, aber nur einmal nach dem Betätigen der Restartfunktion. Damit gibt der Prozessor nach Anweisung der Betriebssoftware seine Arbeitsbereitschaft bekannt. Der Arbeitsspeicher des über den Daten- und Adreßbus angeschlossenen Bildschirmterminals [1] wird, vom IOS-Programm gesteuert, gelöscht. Daraufhin teilt der Computer dem Anwender über das Datensichtgerät seinen Systemnamen sowie einen Stern (Asterix: Zeichen für die Arbeitsbereitschaft) mit. Wird der Stern nicht ausgegeben, so kommen als Fehlerquellen nur noch die beiden PIA-Bausteine und/oder der ACIA (IC 1) in Frage. Mögliche Fehlerquellen sind falsche Schalterstellungen, schlechte Lötstellen, kein Widerstand an Anschluß 25a (zweite VG-Leiste), defekte Bausteine und Leiterbahnfehler.

Systemerweiterung

Zur Kommunikation mit dem Mikrocomuter kann eine ASCII-codierte Tastatur verwendet werden. Sie wird an den PIA, IC 6, zweite VG-Steckerleiste, Anschlüsse 25b...32b, 13b und 14b angeschlossen (14b = Strobe, 32b...26b = DB0...DB6, 13b = Bell).

Die Rechnerausdrucke können über die serielle Schnittstelle des ACIA MC 6850 (IC 7, zweite VG-Steckerleiste, Anschluß 6b) oder über die parallele Datenschnittstelle des PIA (IC 2, zweite VG-Steckerleiste, Anschlüsse14 , 13a, 32a...26a und 25a) entnommen werden (14a = Drucker-Ready, 13a = Rechner-Strobe, 32a...26a = D0...D6, 25a = Drucker-General-Ready).

Zur Überwachung der Computeraktivitäten kann auch über den Daten- und Adreßbus (Bus-VG-Steckerleiste) ein Bildschirmterminal [1] angeschlossen werden. Der Mikrocomputer steuert dazu den 0,5-KByte-Sichtgerätearbeitsspeicher. Das Betriebsprogramm verarbeitet alle Cursor-Steuerbefehle und bestimmt daraufhin den Dateninhalt im Arbeitsspeicher, der im Adressenfeld C000...C1FF liegen muß. Zur Rotierung der Sichtgerätezeilen (jede neue Information wird von unten in den Bildschirm hineingeschoben, die oberste Zeile verlöscht dabei) stehen vier Steuerleitungen an der zweiten VG-Steckerleiste (24b...21b) zur Verfügung (Row-Pointer).

Um den Mikrocomputer auszubauen, kann man über die vordere VG-Steckerleiste einen 16-KByte-Speicher (RAM und/oder ROM) sowie eine Datensichtgerät-Steuerkarte anschließen. Über die zweite VG-Steckerleiste ist mit der seriellen Datenschnittstelle die Verbindung zu einem Kassetten-Interface möglich [2]. Der Anwender kann damit beliebig große Datenmengen aus definierten Speicherblöcken mit einem einfachen Tonbandgerät auf Magnetbändern abspeichern und wieder auslesen und verfügt somit über einen preiswerten Massenspeicher.

Zur EPROM-Programmierung steht dem Anwender eine Programmierkarte zur Verfügung, die für die Speichertypen 2708 (1 KByte) und 2716 (2 KByte) geeignet ist. Oft verwendete Programme lassen sich in die EPROMs eingeben und können jederzeit abgerufen werden, ohne lange Ladezeiten in Kauf nehmen zu müssen.

Um die Erweiterungskarten ohne größere Mühe miteinander verbinden zu können, ist es sinnvoll, eine Busplatine einzusetzen. Auf ihr befinden sich alle Steuer- und Datenleitungen sowie die Adressenzuführungen zwischen den einzelnen VG-Buchsenleisten.

Literatur

[1] *May, U.; May, L.:* Bildschirmterminal für Mikrocomputer. FUNKSCHAU 1978, H. 24, S. 1224...1226, und H. 25, S. 1290...1291.

[2] Datenblatt: Interface 1. 16-KByte-ROM mit Kassetteninterface auf Europakarte, May-Elektronik.

Horst Pelka

8080-Programm zur Periodendauer- und Frequenzmessung

Häufig wird ein Frequenzmesser für niedrige Frequenzen benötigt. Mit einem Programm der Länge von 49 Bytes kann bereits eine Frequenzmessung über die Periodendauer vorgenommen werden. Programm und Hardware sind für den in „Praxis mit Mikroprozessoren" (FUNKSCHAU-Serie bzw. in der Buchreihe „Das kleine Praktikum") beschriebenen Computer ausgelegt.

Bei üblichen Frequenzmessern werden die Null-Durchgänge pro Zeiteinheit gezählt. Dies hat bei tiefen Frequenzen den Nachteil, daß für eine annehmbare Meßgenauigkeit ein sehr langer Meßzeitraum benötigt wird. Will man beispielsweise eine Frequenz von 10 Hz mit einer Genauigkeit von 1 % messen, wird bereits eine Meßzeit von 10 s benötigt. Es ist daher auch bei normalen Frequenzmessern üblich, tiefe Frequenzen nicht direkt, sondern über die Periodendauer zu messen. Mit der nachstehend beschriebenen Anordnung kann man Frequenzen von 1 Hz bis 30 kHz messen, wobei die Meßgenauigkeit mit steigender Frequenz abnimmt. Immerhin ist die Meßgenauigkeit bei 1 kHz besser als 2 %, was in vielen Anwendungsfällen ausreicht. Das Meßergebnis ist nach zwei Perioden bereits verfügbar.

Hardware

Steht die Niederfrequenz bereits als rechteckförmige Spannung mit TTL-Pegeln zur Verfügung, so kann man diese direkt an die Klemme P_{C4} anschalten. Auch ist es möglich, mit Hilfe von Schmitt-Trigger-Bausteinen sich aus der sinusförmigen Nf ein rechteckförmiges Signal zu formen. Der Verfasser benutzte die gleiche Schaltung, welche auch zum Anschluß eines Kassettenrecorders zum Überspielen von Programmen in [2] beschrieben wurde und sich in der Praxis bewährt hat. *Bild 1* zeigt den Schaltungsauszug. Das zu messende Niederfrequenzsignal wird über einen 22-µF-Kondensator auf den Eingang eines CMOS-Inverters geschaltet, zwei weitere Inverter formen das Signal zu einem TTL-Signal mit steilen Anstiegsflanken um. Der 10-kΩ-Einsteller wird so abgeglichen, daß an der Klemme P_{C4} ein rechteckförmiges Signal steht. Dabei verringert man stetig die Eingangsspannung und stellt immer wieder am Potentiometer nach. Der Einstellbereich des Potentiometers, bei dem rechteckförmige Signale an P_{C4} erscheinen, wird immer kleiner, bis man die kleinstmögliche Niederfrequenz erreicht hat, bei der die Meßanordnung noch einwandfrei arbeitet.

Software

Bild 2 zeigt den Programmausdruck. Als erstes wird der Ein-Ausgabebaustein 8255 so programmiert, daß P_{C4} als Eingabe geschaltet ist. Das Registerpaar B, C ist als Schleifenzähler geschaltet. Es wird bei der symbolischen Adresse ABF0 zunächst auf Null gesetzt. Um einen definierten Anfangspunkt für die Abzählung der H- und L-Bereiche zu erzwingen, sind zwei Abfrageschleifen mit den symbolischen Adressen ABF1 und ABF2 vorgesehen. Die Schleife ABF1 wird so lange nicht verlassen, wie H-Pegel an P_{C4} anliegt. Erst dann geht das Programm in die Schleife ABF2 über, so lange, wie L-Pegel an P_{C4} liegt *(Bild 3)*. In Schleife ABF3 wird alle 15,625 µs der Eingangswert der Klemme P_{C4} abgefragt und der Inhalt des Registerpaares B, C um eins erhöht. Geht das Signal auf L über, wird die Schleife ABF3 verlassen und die Anzahl der Abfragen in Schleife ABF4 zu dem bisherigen Inhalt des Registerpaares B, C hinzugefügt. Beim Übergang von P_{C4} auf H-Signal wird Schleife ABF4 verlassen und das höherwertige Byte in die Speicherstelle 00AFH eingeschrieben. Das niederwertige Byte kommt in die Speicherstelle 00AEH. Beim FUNKSCHAU-System [1] kann der Inhalt der Speicherstellen direkt auf der Datenanzeige sichtbar gemacht werden, indem auf dem Modul für direkten Speicherzugriff die Adressen 00AE bzw. 00AF eingestellt werden. Ein Strobe-Impuls schreibt dann das Datenbyte in den Zwischenspeicher des BCD-7-Segmentdecoders ein.

Anzeigeauswertung

Für Frequenzen oberhalb von 250 Hz wird nur das niederwertige Byte des Ergebnisses benötigt. Die angezeigte Hexzahl ist mit 15,625 µs zu multiplizieren und ergibt die Periodendauerlänge. Die Frequenz ist der Reziprokwert der Periodendauer.

In *Bild 4* findet man den Inhalt des Programmspeichers. Die Startadresse wurde hier auf 4800H gesetzt, da dies die Leseadresse der an der Frontplatte zugängigen Fassung für die PROM-Programmiereinrichtung [3] ist. Beim Umschreiben auf eine andere Startadresse sind die Sprungadressen entsprechend zu ändern.

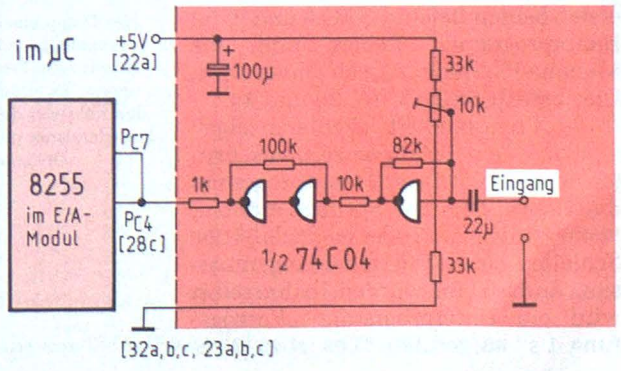


Bild 1. Einfache Vorverstärker- und Schmitt-Trigger-Schaltung zur Verstärkung schwacher Eingangsimpulse

```
00EE          PORTC   EQU     0EEH
4800                  ORG     4800H
4800 3E8A             MVI     A,8AH
4802 D3EF             OUT     0EFH
4804 0600     ABF0:   MVI     B,0
4806 0E00             MVI     C,0
4808 DBEE     ABF1:   IN      PORTC
480A E680             ANI     80H
480C C20848           JNZ     ABF1
480F DBEE     ABF2:   IN      PORTC
4811 E680             ANI     80H
4813 CA0F48           JZ      ABF2
4816 DBEE     ABF3:   IN      PORTC
4818 03               INX     B
4819 E680             ANI     80H
481B C21648           JNZ     ABF3
481E DBEE     ABF4:   IN      PORTC
4820 03               INX     B
4821 E680             ANI     80H
4823 CA1E48           JZ      ABF4
4826 78               MOV     A,B
4827 32AF00           STA     00AFH
482A 79               MOV     A,C
482B 32AE00           STA     00AEH
482E C30448           JMP     ABF0
                      END
```

Bild 2. Programm zum Messen der Periodendauer eines Eingangssignales

Will man die Meßergebnisse direkt in Frequenzen anzeigen, so muß man die Dezimalzahl 64000 = Hexzahl FA00 durch den Inhalt des Registerpaares B, C dividieren. Das Ergebnis ist eine Anzeige der Frequenz in Hz.

Bild 5 zeigt die Liste des Programmspeicherinhaltes. Das Programm beginnt bei der Speicherstelle 4980H und teilt sich folgendermaßen auf:

Hauptprogramm:	4980H bis 49ADH
Unterprogramme:	
Periodenlängenmessung	49AEH bis 49D8H
Fehleranzeige	49D9H bis 4A27H
Division	4A28H bis 4A5FH
Hex-Dezimal-Umwandlung	4A60H bis 4A90H
Ausgabe	4A91H bis 4AB7H
CO (Eingangsadresse 03FAH)	01E3H bis 01EDH

Das Unterprogramm zur Periodenlängenmessung entspricht weitgehend dem Programm nach Bild 2, nur daß die ersten beiden Befehle (SEQ 8 und 9) im Hauptprogramm stehen und die Klemme P_{C7} (nicht P_{C4}) abgefragt wird. Dies betrifft SEQ 13, 16, 20 und 24.

Der Programmteil „Fehleranzeige" gibt den Text „Meßgrenze überschritten" aus. Er wird vom Hauptprogramm durch zwei bedingte Sprünge erreicht, wenn weniger als 3 oder mehr als 64 000 Schleifen bei der Periodenlängenmessung durchlaufen wurden. In ihm selbst wird ein Unterprogramm „Verzögerung 1 s" aufgerufen. Dies ist in [2], S. 188, abgedruckt. Da der Programmteil über Sprungbefehle erreicht wird, endet er auch wieder mit einem Sprungbefehl, der in diesem Fall zum Anfang des Hauptprogramms führt.

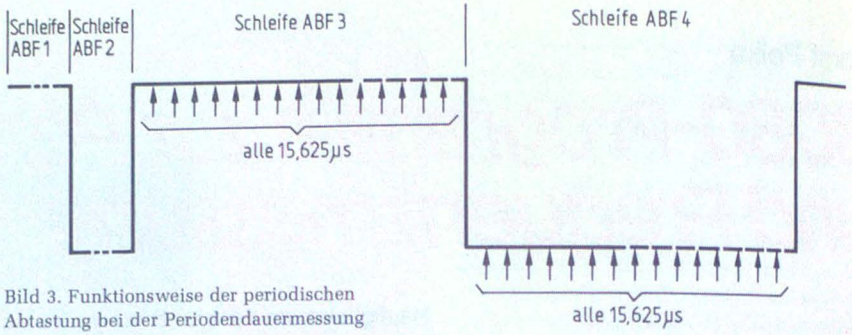


Bild 3. Funktionsweise der periodischen Abtastung bei der Periodendauermessung

Zur Umrechnung der Periodenlänge in die Frequenz wird die Hexzahl 0FA00 (Dezimal 64000) durch die Periodenlänge dividiert. Dies geschieht im Unterprogramm Division. Der Dividend wird im Registerpaar D, E, der Divisior im Registerpaar H, L übergeben. Der Quotient steht nach der Rechenoperation im Registerpaar D, E. Zwischengrößen werden in den RAM-Speicherstellen 1000 H bis 1003 H abgelegt.

Das als Sedezimalzahl vorliegende Ergebnis wird im Unterprogramm „Hex-Dezimal-Umwandlung" in eine fünfstellige Dezimalzahl umgewandelt, die in den RAM-Speicherstellen 1010H bis 1014H zwischengespeichert wird.

Das Unterprogramm „Ausgabe" ruft die einzelnen Stellen ab, wandelt die Zahl in den ASCII-Code um und fügt nach einem Leerzeichen die Bezeichnung Hz hinzu. Am Ende wird nur ein CR-Zeichen (und kein LF) gegeben, damit das alte Meßergebnis von dem neuesten Meßergebnis überschrieben wird. Zur Ausgabe selbst wird das Monitorprogramm CO mit der Einsprungadresse 03FA benutzt, welches intern zur Adresse 01E3 verzweigt und mit abgedruckt ist. Hardwaremäßig erfolgt die Ausgabe über eine serielle Schnittstelle, die in [2] ab S. 58 beschrieben ist.

Literatur

[1] *Pelka, H.*: Praxis mit Mikroprozessoren. FUNKSCHAU 1977, Hefte 15–26, und 1978, Hefte 1–10.

[2] *Pelka, H.*: Praxis mit Mikroprozessoren. Das kleine Praktikum, Franzis-Verlag, Seite 182–189.

[3] *Pelka, H.*: PROM-Programmier-Einrichtung benötigt nur 36 Byte an Software. Elektronik 1978, Heft 9, Seite 97–101.


```
4800 54 01 00 00 C5 01 AA 47 C5 11 C8 57 01 01 00 CD
4810 E1 54 2A C3 57 44 4D CD 9F 4D 3A C7 57 1F D2 24
4820 48 CD 46 54 C9 21 CB 57 70 2B 71 2A CA 57 7E FE
4830 0D
```

Bild 4. Hex-Dump der Routine zum Anzeigen des Meßergebnisses. Die angezeigte Hexzahl ist mit 15,625 zu multiplizieren und ergibt die Periodendauer in Mikrosekunden

```
4980 3E 8A D3 EF CD AE 49 79 32 AE 00 FE 03 DA D9 49
4990 78 32 AF 00 D6 FA D2 D9 49 60 69 16 FA 1E 00 CD
49A0 28 4A 21 10 10 CD 60 4A CD 91 4A C3 84 49 06 00
49B0 0E 00 DB EE E6 80 C2 B2 49 DB EE E6 80 CA B9 49
49C0 DB EE 03 E6 80 C2 C0 49 DB EE 03 E6 80 CA C8 49
49D0 78 32 AF 00 79 32 AE 00 C9 21 F2 49 4E CD FA 03
49E0 7E FE 0D 23 C2 DC 49 FE 03 CA EF 49 CD DF 41 C3
49F0 80 49 4D 45 53 53 47 52 45 4E 5A 45 20 55 45 42
4A00 45 52 53 43 48 52 49 54 54 45 4E 0A 0D 20 20 20
4A10 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
4A20 20 20 20 20 20 20 0D 03 22 00 10 01 00 00 C5 21
4A30 02 10 36 11 7B 17 5F 7A 17 57 21 02 10 35 E1 CA
4A40 59 4A 3E 00 17 29 44 85 2A 00 10 95 4F 78 9C 47
4A50 C5 D2 34 4A 09 E3 C3 34 4A 7A 2F 57 7B 2F 5F C9
4A60 01 04 00 09 EB 01 F0 D8 CD 81 4A 01 18 FC CD 81
4A70 4A 01 9C FF CD 81 4A 01 F6 FF CD 81 4A 7D 12 EB
4A80 C9 AF D5 5D 54 3C 09 DA 83 4A 3D 6B 62 D1 12 1B
4A90 C9 01 04 00 09 1E 05 7E C6 30 4F CD FA 03 2B 1D
4AA0 C2 97 4A 0E 20 CD FA 03 0E 48 CD FA 03 0E 5A CD
4AB0 FA 03 0E 0D CD FA 03 C9 FF FF FF FF FF FF FF FF
01E0 FF FF FF DB FB E6 01 CA E3 01 79 D3 FA C9 FF FF
```

Bild 5. Hex-Dump eines Programms zur Anzeige der Frequenz. Es bildet den Kehrwert der Periodendauer per Division

Norbert Ulamec

TI-59 dimensioniert Verstärker

Das nachfolgende Taschenrechnerprogramm erlaubt die überschlägige Dimensionierung der Schaltelemente eines einstufigen Verstärkers in Emitterschaltung mit kapazitiv angekoppelter Last. Dabei wird auch der günstigste Kompromiß zwischen Aussteuerbarkeit und Stromverbrauch errechnet.

Ein Verstärker ist für symmetrische Signale dann optimal dimensioniert, wenn er bei geringstmöglichem Stromverbrauch die geforderte Ausgangsleistung am Lastwiderstand gerade abgeben kann, ohne daß Signalverzerrungen auftreten. Bei Übersteuerungen des Verstärkers soll die Signalbeschneidung symmetrisch sein, d. h. gleichmäßig an den oberen und unteren Spitzen. Diese Betriebsart ist der sogenannte „A-Betrieb". Für die Dimensionierung wurde die Emitterschaltung wegen ihrer guten Leistungsverstärkung gewählt, *Bild 1* zeigt die Standardsschaltung.

Das Programm *(Bild 2)* kann auf dem Rechner TI-59 angewendet werden. Für den Verstärker können alle Bauteile mit Ausnahme des Transistors, der aus einem Datenbuch ermittelt wird, berechnet werden.

Die errechneten Werte werden bereits in der E-12-Normreihe ausgegeben und bei Verwendung eines Druckers (PC 100 B/A) auf dem Ausdruck festgehalten. Die Kondensatoren werden nicht genormt, da man den tatsächlichen Wert deutlich größer als den errechneten Wert wählen soll.

Vor dem Eingeben des Programms ist es wichtig, die Speicherbereichsverteilung auf 559/49 zu ändern. Die zur Aufzeichnung erforderlichen Magnetkarten können ohne Änderung der Bereichsverteilung nicht wieder eingelesen werden.

Da der Rechner unter dem Programmablauf eine Datenkarte mit den Normwerten anfordert, ist es zweckmäßig, die Werte von Widerständen und Batteriespannungen *(Tabelle)* auf einer Magnetkarte unter Block 4 zu speichern.

Die Bedienung geht wie folgt vor sich: Die Taste A am Rechner wird gedrückt; ist der Rechner auf den Drucker gesetzt, so wird die Überschrift gedruckt. Als nächstes fordert der Rech-

Tabelle: Inhalt der Datenregister

Nr.	Wert	Nr.	Wert	Nr.	Wert
0	0	10	4,7	20	18
1	0	11	5,6	21	24
2	1	12	6,8	22	32
3	1,2	13	8,2	23	48
4	1,5	14	10	24	100
5	1,8	15	5		
6	2,2	16	6		
7	2,7	17	8		
8	3,3	18	12		
9	3,9	19	15		

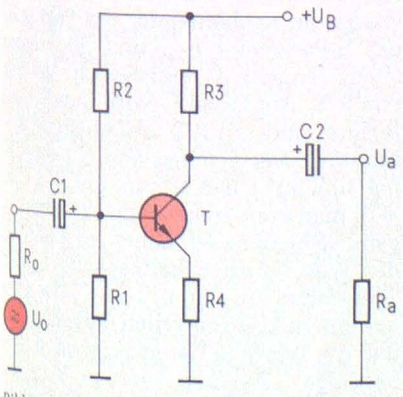


Bild 1. Standardschaltung für einen einfachen Verstärker

```
PROGRAMMAUFLISTUNG
EMITTERSCHALTUNG

000  76 LBL
001  11  A
002  69 OP
003  00  00
004  01  1
005  07  7
006  04  4
007  00  0
008  00  0
009  00  0
010  03  3
011  06  6
012  04  4
013  00  0
014  69 OP
015  01  01
016  69 OP
017  05  05
018  98 ADV
019  04  4
020  22 INV
021  96 WRT
022  00  0
023  91 R/S
024  76 LBL
025  12  B
026  48 EXC
027  25  25
028  48 EXC
029  29  29
030  48 EXC
031  28  28
032  48 EXC
033  27  27
034  48 EXC
035  26  26
036  91 R/S
037  76 LBL
038  13  C
039  00  0
040  32 X:T
041  43 RCL
042  10  10
043  67  EQ
044  54  )
045  25 CLR
046  04  4
047  01  1
048  01  1
049  04  4
050  69 OP
051  04  04
052  25 CLR
053  01  1
054  05  5
055  42 STO
056  00  00
057  43 RCL
058  27  27
059  55  ÷
060  43 RCL
061  26  26
062  95  =
063  42 STO
064  30  30
065  02  2
066  93  .
067  05  5
068  85  +
069  02  2
070  55  ÷
071  43 RCL
072  30  30
073  95  =
074  65  ×
075  43 RCL
076  27  27
077  85  +
078  43 RCL
079  29  29
080  95  =
081  32 X:T
082  76 LBL
083  22 INV
084  73 RC*
085  00  00
086  77  GE
087  23 LNX
088  69 OP
089  20  20
090  61 GTO
091  22 INV
092  76 LBL
093  23 LNX
094  42 STO
095  31  31
096  69 OP
097  06  06
098  91 R/S
099  76 LBL
100  14  D
101  25 CLR
102  03  3
103  05  5
104  00  0
105  04  4
106  69 OP
107  04  04
108  71 SBR
109  45 Y^x
110  43 RCL
111  31  31
112  75  -
113  43 RCL
114  29  29
115  95  =
116  55  ÷
117  43 RCL
118  27  27
119  75  -
120  02  2
121  75  -
122  02  2
123  55  ÷
124  43 RCL
125  30  30
126  95  =
127  65  ×
128  43 RCL
129  28  28
130  95  =
131  71 SBR
132  24 CE
133  32 X:T
134  76 LBL
135  32 X:T
136  73 RC*
137  01  01
138  22 INV
139  77  GE
140  33 X²
141  69 OP
142  31  31
143  61 GTO
144  32 X:T
145  76 LBL
146  33 X²
147  65  ×
148  43 RCL
149  34  34
150  95  =
151  42 STO
152  32  32
153  69 OP
154  06  06
155  91 R/S
156  76 LBL
157  15  E
158  25 CLR
159  03  3
160  05  5
161  00  0
162  05  5
```

Bild 2. Auflistung des TI-59-Programms. Die Speicherverteilung ist auf 559/49 zu ändern ▶

163	69	DP	244	65	×	325	69	DP	406	95	=	487	42	STO
164	04	04	245	43	RCL	326	06	06	407	71	SBR	488	47	47
165	01	1	246	37	37	327	91	R/S	408	24	CE	489	02	2
166	00	0	247	95	=	328	76	LBL	409	32	X:T	490	01	1
167	42	STO	248	94	+/-	329	18	C'	410	76	LBL	491	69	DP
168	34	34	249	85	+	330	25	CLR	411	52	EE	492	04	04
169	02	2	250	43	RCL	331	03	3	412	73	RC*	493	43	RCL
170	42	STO	251	31	31	332	05	5	413	01	01	494	28	28
171	01	01	252	95	=	333	00	0	414	22	INV	495	35	1/X
172	43	RCL	253	42	STO	334	02	2	415	77	GE	496	85	+
173	32	32	254	38	38	335	69	DP	416	53	(	497	43	RCL
174	35	1/X	255	69	DP	336	04	04	417	69	DP	498	32	32
175	85	+	256	06	06	337	71	SBR	418	31	31	499	35	1/X
176	43	RCL	257	91	R/S	338	45	Y×	419	61	GTO	500	95	=
177	28	28	258	03	3	339	43	RCL	420	52	EE	501	35	1/X
178	35	1/X	259	03	3	340	40	40	421	76	LBL	502	65	×
179	95	=	260	01	1	341	65	×	422	53	(	503	43	RCL
180	35	1/X	261	05	5	342	43	RCL	423	65	×	504	47	47
181	42	STO	262	69	DP	343	41	41	424	43	RCL	505	95	=
182	36	36	263	04	04	344	65	×	425	34	34	506	35	1/X
183	55	÷	264	43	RCL	345	43	RCL	426	95	=	507	55	÷
184	43	RCL	265	37	37	346	42	42	427	42	STO	508	02	2
185	30	30	266	65	×	347	55	÷	428	44	44	509	55	÷
186	95	=	267	43	RCL	348	43	RCL	429	69	DP	510	89	π
187	71	SBR	268	38	38	349	37	37	430	06	06	511	95	=
188	24	CE	269	95	=	350	55	÷	431	91	R/S	512	69	DP
189	32	X:T	270	69	DP	351	53	(	432	76	LBL	513	06	06
190	76	LBL	271	06	06	352	01	1	433	19	D'	514	91	R/S
191	35	1/X	272	91	R/S	353	75	-	434	01	1	515	76	LBL
192	73	RC*	273	04	4	354	43	RCL	435	05	5	516	45	Y×
193	01	01	274	01	1	355	42	42	436	00	0	517	01	1
194	77	GE	275	01	1	356	55	÷	437	02	2	518	00	0
195	42	STO	276	04	4	357	43	RCL	438	69	DP	519	42	STO
196	69	DP	277	00	0	358	31	31	439	04	04	520	34	34
197	21	21	278	01	1	359	54	)	440	02	2	521	01	1
198	61	GTO	279	69	DP	360	95	=	441	65	×	522	03	3
199	35	1/X	280	04	04	361	71	SBR	442	89	π	523	42	STO
200	76	LBL	281	43	RCL	362	24	CE	443	65	×	524	01	01
201	42	STO	282	37	37	363	32	X:T	444	43	RCL	525	92	RTN
202	65	×	283	65	×	364	76	LBL	445	45	45	526	76	LBL
203	43	RCL	284	43	RCL	365	43	RCL	446	95	=	527	24	CE
204	34	34	285	35	35	366	73	RC*	447	42	STO	528	42	STO
205	95	=	286	85	+	367	01	01	448	46	46	529	33	33
206	42	STO	287	43	RCL	368	22	INV	449	65	×	530	01	1
207	35	35	288	25	25	369	77	GE	450	53	(	531	00	0
208	69	DP	289	95	=	370	44	SUM	451	43	RCL	532	32	X:T
209	06	06	290	42	STO	371	69	DP	452	43	43	533	76	LBL
210	91	R/S	291	39	39	372	31	31	453	35	1/X	534	34	fX
211	76	LBL	292	69	DP	373	61	GTO	454	85	+	535	43	RCL
212	16	A'	293	06	06	374	43	RCL	455	43	RCL	536	33	33
213	02	2	294	91	R/S	375	76	LBL	456	44	44	537	55	÷
214	04	4	295	76	LBL	376	44	SUM	457	35	1/X	538	01	1
215	01	1	296	17	B'	377	65	×	458	54	)	539	00	0
216	05	5	297	48	EXC	378	43	RCL	459	35	1/X	540	95	=
217	69	DP	298	45	45	379	34	34	460	95	=	541	77	GE
218	04	04	299	48	EXC	380	95	=	461	35	1/X	542	25	CLR
219	43	RCL	300	41	41	381	42	STO	462	69	DP	543	92	RTN
220	27	27	301	48	EXC	382	43	43	463	06	06	544	76	LBL
221	55	÷	302	40	40	383	69	DP	464	91	R/S	545	25	CLR
222	43	RCL	303	91	R/S	384	06	06	465	01	1	546	42	STO
223	36	36	304	04	4	385	91	R/S	466	05	5	547	33	33
224	95	=	305	01	1	386	03	3	467	00	0	548	01	1
225	42	STO	306	01	1	387	05	5	468	03	3	549	00	0
226	37	37	307	04	4	388	00	0	469	69	DP	550	49	PRD
227	69	DP	308	00	0	389	03	3	470	04	04	551	34	34
228	06	06	309	01	1	390	69	DP	471	43	RCL	552	61	GTO
229	91	R/S	310	06	6	391	04	04	472	32	32	553	34	fX
230	04	4	311	05	5	392	71	SBR	473	85	+	554	76	LBL
231	01	1	312	69	DP	393	45	Y×	474	43	RCL	555	54	)
232	01	1	313	04	04	394	43	RCL	475	28	28	556	00	0
233	05	5	314	01	1	395	43	43	476	95	=	557	23	LNX
234	01	1	315	85	+	396	65	×	477	65	×	558	91	R/S
235	07	7	316	43	RCL	397	53	(	478	43	RCL	559	00	0
236	69	DP	317	40	40	398	43	RCL	479	46	46			
237	04	04	318	95	=	399	31	31	480	95	=			
238	43	RCL	319	65	×	400	55	÷	481	35	1/X			
239	32	32	320	43	RCL	401	43	RCL	482	69	DP			
240	85	+	321	39	39	402	42	42	483	06	06			
241	43	RCL	322	95	=	403	75	-	484	91	R/S			
242	35	35	323	42	STO	404	01	1	485	76	LBL			
243	95	=	324	42	42	405	54	)	486	10	E'			

```
          0.75      UE
           7.5      UA
          2200.     RA
           0.5      USAT
           0.6      UBE
          100.      F



           24.      UB
          1800.     R3
          100.      R4
 .0075757576        IC
9.606060606         UCE
.0727731864         PC
1.357575758         UBO
1.493333333         UBO'
          6800.     R1
        100000.     R2
 0.00000025         C1
.0000003979         C2
8038128.439         F
```

Bild 3. Typischer Ausdruck beim Errechnen der Werte für einen Nf-Vorverstärker. Es wurde mit folgenden Angaben gerechnet: $\hat{U}_e = 0{,}75$ V, $\hat{U}_a = 7{,}5$ V, $R_a = 2200\ \Omega$, $f_u = 100$ Hz (untere zu übertragende Frequenz)

ner die Datenkarte an. Nach der Eingabe erscheint 0 in der Anzeige, und die manuelle Dateneingabe auf Label B kann erfolgen. Die Daten von Eingangsspannung U_e, Ausgangsspannung U_a, Lastwiderstand R_a, Sättigungsspannung U_{sat} und Basis-Emitter-Spannung U_{BE} werden nacheinander auf B eingegeben. Die Batteriespannung U_B errechnet sich auf Label C, der Widerstand R 3 auf Label D und der Widerstand R 4 auf Label E. Auf Label A' wird zuerst der Kollektorstrom I_C errechnet; drückt man nun mehrmals R/S, so erscheinen die Werte von Kollektor-Emitter-Spannung U_{CE}, Leistung P_C des Transistors und die Basisspannung gegen Masse U_{BO}. Jetzt muß eine neue Dateneingabe, nämlich jene des ausgewählten Transistors, erfolgen. Auf Label B' werden wiederum nacheinander der Faktor α, der h-Parameter h_{21e} und die untere Grenzfrequenz f_u eingegeben, gefolgt von R/S. Auf Label C' wird nun R 1 errechnet und mit R/S anschließend R 2 ermittelt. Der Kondensator C 1 errechnet sich auf Label D' und mit R/S C 2. Will man noch die maximal zu übertragende Frequenz errechnen, so gibt man den Wert für die Schaltkapazität C_a des Transistors ein und drückt E'. Das Programm ist kontinuierlich aufgebaut, so daß die Werte in der angegebenen Reihenfolge errechnet werden müssen. *Bild 3* zeigt ein Beispiel einer typischen Vorverstärker-Stufe.

Alles auf einer Platine

Mikrocomputer für die Programmierung in Maschinensprache

Wer sich in die Mikrocomputer-Technik einarbeiten möchte, wäre mit einem der heute bereits recht preiswert erhältlichen BASIC-Systeme schlecht beraten: Diese Systeme erlauben es meist nicht ohne weiteres, auf die „Maschinenebene" herabzusteigen und Programme in der CPU-Sprache hexadezimal oder im Assembler-Code einzugeben. Dafür sind spezielle Einplatinen-Computer besser geeignet; sie lassen sich später erweitern, so daß – wenn dies gewünscht wird – auch BASIC implementiert werden kann.

Unsere *Tabelle* auf Seite 1368 und 1369 enthält die meisten heute erhältlichen Mikrocomputer, die in erster Linie für das Programmieren in Maschinensprache konzipiert sind. Sie entspricht dem Stand vom August 1979; Änderungen sind natürlich vor allem in den Spalten „Preis" und „Vertrieb" möglich. Bei den Preisen sind ganz erhebliche Schwankungen bei unterschiedlichen Händlern festzustellen, so daß die angegebenen Kosten nur als Richtwert betrachtet werden können.

Die meisten Hersteller sind bemüht, eine ständig wachsende Zahl an Erweiterungs-Baugruppen und Software anzubieten. Es ist also damit zu rechnen, daß für alle aufgeführten Computer im Laufe der Zeit auch Erweiterungen lieferbar sein werden, die in der Tabelle noch nicht aufgeführt sind.

Kassetten-Interface

Ein Teil der in der Tabelle aufgeführten Mikrocomputer verfügt über ein Interface zum Aufzeichnen von Programmen auf einem normalen Kassettenrecorder. Die Platinen AIM-65, HB-61000, ECB-85, TM-990/189 und Eurocom-1 arbeiten dabei mit der Kansas-City-Norm, d.h. den Tonfrequenzen 1200 Hz (L) und 2400 Hz (H). Das Datenformat ist dadurch allerdings noch nicht festgelegt und tatsächlich bei den meisten Mikrocomputern untereinander nicht kompatibel (außer AIM, KIM und SYM).

Manche Computer-Platinen verfügen über kein Kassetten-Interface, sondern „nur" über eine RS-232-Schnittstelle bzw. eine Stromschleifen-Schnittstelle, an die sich außer einem Terminal auch ein Kassetten-Modem anschließen läßt, wie es bereits in FUNKSCHAU 1978, Heft 14 und 15, beschrieben wurde. Der zusätzliche Aufwand beträgt bei dieser Schaltung nur ein Standard-CMOS-IC und drei Transistoren, so daß das Fehlen einer Kassetten-Interface-Schaltung auf der Computerplatine kein allzu großes Manko darstellt.

Serielle Schnittstelle

Zahlreiche Computer können über eine serielle Schnittstelle mit einem ASCII-Fernschreiber oder einem Videoterminal korrespondieren. Die Bezeichnung der Schnittstelle mit RS-232 bzw. V-24 und 20-mA- bzw. 40-mA-Stromschleife gibt lediglich die Pegelverhältnisse auf dieser Leitung für log. 0 und log. 1 an. Das Datenformat ist jedoch weitgehend genormt (asynchroner Start-Stop-Betrieb mit oder ohne Parity-Bit). Der Computer kann sich dabei auf die vom Terminal benutzte Übertragungsgeschwindigkeit automatisch einstellen, wenn die Seriell/Parallel-Wandlung per Software und nicht mit einem UART geschieht; beim Nascom-1 ist die automatische Anpassung zum Beispiel nicht möglich, da er mit einem festen UART-Takt arbeitet.

Erweiterbarkeit

Irgendwie ist jedes Mikrocomputersystem erweiterbar. Einschränkungen bestehen in dreierlei Hinsicht: Erstens muß natürlich die physikalische Voraussetzung für den Anschluß zusätzlicher Bausteine an den Adressen- und Datenbus gegeben sein. Das bedeutet, daß irgendeine Steckleiste vorhanden sein muß, an die diese Leitungen geführt sind. Dies ist praktisch bei allen Mikrocomputern der Fall. Bei Erweiterungen, die eine alphanumerische Ein- und Ausgabe erfordern, wird es schon schwieriger; eine ASCII- bzw. Buchstaben- und Ziffern-Tastatur nebst einem alphanumerischen Display oder Video-Interface ist Voraussetzung für das Arbeiten mit einem Assembler oder in BASIC. Der TM 990/189 ist hier ein Kompromiß; seine Tastatur enthält zwar alle Buchstaben und Ziffern, aber nicht in Schreibmaschinenanordnung, sondern taschenrechner-ähnlich, was die Bedienbarkeit ebenso erschwert wie das Siebensegment-Display, auf dem die Buchstaben nur sehr stilisiert dargestellt werden können. Eine dritte Voraussetzung für die Erweiterbarkeit ist natürlich, überhaupt Erweiterungen zu bekommen, sei es nun Software oder Hardware. Für die Erhältlichkeit von Programmen ist ein Blick in die Mikrocomputer-Seiten der FUNKSCHAU ein guter Anhaltspunkt; was die Hardware angeht, sollte man sich an den Hersteller selbst wenden. Fe.

Ein neuer Apple

Eine verbesserte Version des bekannten Apple-II wird die Apple Computer Co. in Kürze unter der Bezeichnung Apple Plus liefern. Während das bisherige Modell den Applesoft-Basic-Interpreter erst von der Kassette laden mußte, steht jetzt das Basic im ROM sofort nach dem Einschalten zur Verfügung. Der Reset-Knopf ist zwar zugänglich, führt aber nicht zu einer Zerstörung des gerade im RAM stehenden Basic-Programms. Ähnlich wie beim PET 2001 ist ein komfortables Redigieren auf dem Bildschirm möglich. Der Apple Plus ist in 16-, 32- und 48-KByte-Versionen (RAM) erhältlich.

Typ	KIM-1	AIM-65	ET-3400	SDK-85-Kit	SDK-86-Kit	Eurocom-1	Nascom-1	SYM-1	TM 990/189	F387X-PEP	ECB-85
Hersteller	Commodore	Rockwell	Heathkit	Intel	Intel	Eltec	N.A.S.	Synertek	Texas Instruments	Fairchild	Siemens
Vertrieb	Neumüller, Eschenstr. 2, 8021 Taufkirchen	GWK, Asternstr. 2, 5120 Herzogenrath Feltron, 5210 Troisdorf-Spich	Heathkit, Robert-Bosch-Str. 32-38, 6072 Dreieich	Electronic 2000, Neumarkter Str. 75, 8000 München 80	Electronic 2000, Neumarkter Str. 75, 8000 München 80	Eltec, Postfach 1847, 6500 Mainz	N.A.S., Am Weidengraben 74, 5500 Trier	Astronic Winzererstr. 47d, 8000 München 40	Texas Instruments Deutschland Haggertystr. 1, 8050 Freising	u.a. Electronic 2000, Neumarkter Str. 75, 8000 München 80	Siemens-µC-Vertrieb, Balanstr. 73, 8000 München 80
Preis (inkl. MwSt., ca.)	530 DM	1000 DM	600 DM	785 DM	2164 DM	390 DM	1000 DM	800 DM	902 DM	1110 DM	1580 DM
CPU-Typ	6502	6502	6800	8085	8086	6802	Z-80	6502	TMS 9980 A	3850	8085
ROM-Monitor/KByte	2	8	1	2	8	2	1	4	4	2	2
ausbaubar auf K/Byte	–	20	–	4	–	–	2	8	6	8	4
RAM/KByte	1	1	0,256	0,256	2	1	1	1	1	2	1
ausbaubar auf/KByte	–	4	4	0,512	4	–	–	4	2	4	1,256
Timer	2	4	–	1	–	–	–	8	2	4	2
RS-232-Schnittstelle bzw. Stromschleife	ja	ja	Option	ja	ja	Option	ja	ja	Option	ja	–
Eingabe	Hex-Tastatur	ASCII-Tastatur	Hex-Tastatur	Hex-Tastatur	Hex-Tastatur	Hex-Tastatur	ASCII-Tastatur	Hex-Tastatur	Bu.-u.Zi.-Tasten in TI-59-Form	Hex-Tastatur	Hex-Tastatur
Ausgabe	Siebensegment-Display	20st.16-Segment-Display	Siebensegment-Display	Siebensegment-Display	Siebensegment-Display	Siebensegment-Display	Video-Ausgang	Siebensegment-Display	Siebensegment-Display	Siebensegment-Display	Siebensegment-Display
E/A-Leitungen (frei)	15	16	8	38	48	16	16	51	16	8	44
Kassetten-Interface	ja	ja (2)	Option	–	–	ja	ja	ja (2)	ja	–	ja
Platinengröße (mm)	300 x 230	305 x 270	295 x 290	305 x 254	343 x 305	232 x 160	300 x 210	273 x 210	280 x 205	335 x 190	230 x320
Betriebssystem: Einzelschritt-Betrieb	ja	ja	ja	ja	ja	ja	ja	ja	ja	ja	ja
Assembler/ Disassembler	–	ja	–	–	–	–	Kassette 160DM	ROM-Option	ja	–	–
Texteditor	–	ja	–	–	–	–	–	–	ja	–	–
BASIC	–	ROM-Option	ROM-Option	–	–	–	Kassette 180DM	ROM-Option	–	–	–
Weitere Besonderheiten	Extrem viel Software erhältlich	Drucker auf der Platine	Sehr umfangreiches Begleitmaterial		16-bit-CPU				16-bit-CPU	Mit EPROM-Programmiersockel + Emulationskabel	Mit EPROM-Programmiersockel

Typ	Alpha-1	Z-8000-DM	HB-61000	Z-80-KIT	MCT-48-2E	Instructor 50	PC 9500	FUNKSCHAU	ELDO	Eurocom-9	OEM-80E
Hersteller	MCS	Kontron	Harris	Kontron	Valvo	Valvo	Valvo	Bauanleitung	ELO-Bauanleitung	Eltec	Mostek
Vertrieb	MCS, Zentral-flughafen Tempelhof, 1000 Berlin 42	Kontron, Breslauer Str. 2, 8057 Eching	Kontron, Breslauer Str. 2, 8057 Eching	Kontron, Breslauer Str. 2, 8057 Eching	Valvo, Postfach 10 63 23, 2000 Hamburg 1	Valvo, Postfach 10 63 23, 2000 Hamburg 1	Valvo, Postfach 10 63 23, 2000 Hamburg 1	Platine und Bausatz May, Schlosserbreite 6, 8018 Grafing	Unitronic, Münsterstr. 338, 4000 Düsseldorf 30	Eltec, Postfach 1847, 6500 Mainz	Alfred Neye Schillerstr. 14, 2085 Quickborn
Preis (inkl. MwSt., ca.)	1100 DM	3616 DM	1650 DM	789 DM	2580 DM	1170 DM	500 DM	je nach Bauel.	je n. Ausbaust.	1080 DM	1811 DM
CPU-Typ	6502	Z-8002	HM-6100	Z-80	8048	2650	2650	6802	2650	6809	Z-80
ROM-Monitor/KByte	2	4	1K x 12 bit	1	1	2	1	1	0,256	4	–
ausbaubar auf K/Byte	–	16	–	4	–	–	24	5	1,512	16	20
RAM/KByte	1,25	32	256 x 12 bit	0,256	2	0,64	0,5	1,125	1,25	16	4
ausbaubar auf/KByte	–	64	1 K x 12 bit	–	–	–	–	–	–	48	16
Timer	2	4	1	–	1	–	–	–	–	–	4
RS-232-Schnittstelle bzw. Stromschleife	ja	ja (2)	ja	ja	ja	ja	ja	ja	Option	ja	ja
Eingabe	Hex-Tastatur	Hex-Tastatur	Oktal-Tastatur	Hex-Tastatur	Hex-Tastatur	Hex-Tastatur	via Schnittstelle	via Schnittstelle	Hex-Tastatur	via Schnittstelle	via Schnittstelle
Ausgabe	Siebensegment-Display	Siebensegment-Display	Siebensegment-Display	Siebensegment-Display	Siebensegment-Display	Siebensegment-Display	via Schnittstelle	via Schnittstelle	Siebensegment-Display	Video-Ausgang	via Schnittstelle
E/A-Leitungen (frei)	16	32	12	16	27	8	16	16	Option 48	32	32
Kassetten-Interface	ja	–	ja	ja	–	ja	–	–	Option	ja	–
Platinengröße (mm)	100 x160	300 x 200	210 x 290	160 x 100	230 x 250	420 x 240	173 x 201	100 x 160	250 x 250	160 x 232	250 x 233
Betriebssystem: Einzelschritt-Betrieb	ja	ja	ja	ja	ja	ja	–	ja	ja	ja	ja
Assembler/ Disassembler	Einf. Dissas	–	–	–	Option	–	–	–	Option	Option	ROMs: 714DM
Texteditor	–	–	–	–	ja	–	ja	–	–	–	Im Assembler
BASIC	–	–	–	ROM-Option	–	–	–	–	–	Option	–
Weitere Besonderheiten	mit Gehäuse	16-bit-CPU	12-bit-CPU		8048-Entwicklungs-System	mit Gehäuse		s. FUNKSCHAU 21...23/1979	Modular ausbaufähig s. ELO ab 9/78		

Superhirn

Das in Basic geschriebene Superhirn-Programm ermöglicht es, ohne Spielpartner zu spielen, und belegt einen Speicherplatz von 1563 Byte. Nach dem Start stellt es sich selbst vor: Es werden die Spielregeln ausgegeben. Anschließend erfolgt die Eingabe für die gewünschte Ziffernlänge. Hierbei kann zwischen einer vier- und fünfstelligen Zahl gewählt werden. Es folgt die Ausgabe des Spielfeldes.

In den Zeilen 13...16 werden die einzelnen Ziffern der Zufallszahl ermittelt. Hierbei wird die Zeit „TI" als Zufallsgröße in den Befehl „RND" eingebracht. Die Zeile 16 stellt sicher, daß die Ziffern immer im Bereich von 1...9 liegen. Nun folgt ein Sprung zur Zeile 1000. Hier wird sichergestellt, daß keine Ziffer mehrfach gespeichert wird. Mit dem Befehl „ON X GOTO" werden die einzelnen Ziffern in die Variablen AA...EE abgespeichert. Anschließend erfolgt der Rücksprung zur Zeile 20. Hier wird verglichen, ob alle Ziffern abgespeichert sind. Ist dies der Fall, so kann die Eingabe eines Rateversuches erfolgen. Dies geschieht in Zeile 25 in der Form eines Strings. Zeile 26 stellt den Versuchszähler dar. Die Zeilen 27 oder 28 bewirken einen Abbruch des Spiels mit der Ausgabe der zu ermittelnden Zahl.

In den Zeilen 31...35 wird der String in einzelne Zahlvariablen umgewandelt. Ab Zeile 500 werden der Versuch ausgewertet und die jeweiligen Zähler betätigt. In dem Moment, in dem der Zähler „J" mit der Variablen „K" identisch ist, wird das Spiel beendet und das Ergebnis ausgegeben. Wenn der Vergleich „J – K" negativ ausfällt, werden die Variablen „V,J,N" ausgegeben und „J" und „N" auf Null gesetzt. Es erfolgt ein Sprung zur Zeile 25, wo eine neue Eingabe erfolgt. Die Zeilen 2002...2004 geben Auskunft über das Endergebnis.

Verzeichnis der Variablen

Variable	Funktion
K	legt die Ziffernlänge fest
A	enthält die einzelnen Ziffern der Zufallszahl
X	dient zum berechneten Sprung
AA...EE	enthalten die Zufallszahl
A$	Eingangsvariable
V	Zähler für die Anzahl der Versuche
A1...A5	enthalten die Ziffern der Eingabe
J	Zähler für die Anzahl der richtigen Ziffern am richtigen Platz
N	Zähler für die Anzahl der richtigen Ziffern am falschen Platz

Die Zeilen 2004...2006 ermöglichen ein neues Spiel, ohne das Programm neu starten zu müssen.

Die Eingabe erfolgt als String und wird in vier oder fünf Zahlvariablen umgewandelt. Dieser Weg wurde beschritten, um den PET 2001 nicht in die Verlegenheit zu bringen, eine vier- oder fünfstellige Zahl mittels mathematischer Befehle in einzelne Ziffern umwandeln zu müssen, da er bekanntlich nicht der genaueste Rechner im Nachkommabereich ist.

Leider kommt es leicht vor, daß sich bei der manuellen Programmeingabe Fehler einschleichen. Um nun das Programm überprüfen zu können, kann man folgende Zeile einschreiben:

21 PRINT AA;BB;CC;DD;EE

Hier wird nun das richtige Ergebnis sofort eingeblendet. Bei einer vierstelligen Zahl wird die fünfte Stelle als Null ausgegeben; sie bleibt dann unberücksichtigt.

Walter Dochow

```
1 PRINT"■"
2 PRINT"R S U P E R H I R N ="
3 PRINT:PRINT"DU MUST EINE VIER- ODER FUENFSTELLIGE   ZAHL ERRATEN.";
4 PRINT" JEDE ZIFFER KOMMT NUR EIN-MAL VOR."
5 PRINT"SOLLTEST DU DIE ZAHL NICHT ERRATEN, SO  KANNST DU SIE DURCH",
6 PRINT"EINGABE DES ZEICHENSR @ = ABRUFEN."
10 CLR:PRINT"SOLL DIE ZAHL R 4 = ODER R 5 = ZIFFERN HABEN" :INPUTK:PRINT"■"
11 PRINT"     RVERSUCH   NR.  PLATZ    ZAHL="
13 FORX=1TOK
15 A=INT(10*(RND(TI)))
16 IFA<0ORA>9THEN15
19 GOTO1000
20 NEXT
25 PRINT:INPUT"     ";A$
26 V=(V+1)
27 IFK=4ANDA$="@"THENPRINT"R",AA;BB;CC;DD"=":PRINT:GOTO2004
28 IFK=5ANDA$="@"THENPRINT"R",AA;BB;CC;DD;EE"=":PRINT:GOTO2004
31 AA$=LEFT$(A$,1):BB$=MID$(A$,2,1)
32 CC$=MID$(A$,3,1):DD$=MID$(A$,4,1)
33 EE$=RIGHT$(A$,1):A5=VAL(EE$)
34 A1=VAL(AA$):A2=VAL(BB$)
35 A3=VAL(CC$):A4=VAL(DD$)
40 GOTO500
500 IFA1=AATHENGOSUB600
501 IFA1=BBORA1=CCORA1=DDTHENGOSUB700
502 IFA2=BBTHENGOSUB600
503 IFA2=AAORA2=CCORA2=DDTHENGOSUB700
504 IFA3=CCTHENGOSUB600
505 IFA3=AAORA3=BBORA3=DDTHENGOSUB700
506 IFA4=DDTHENGOSUB600
507 IFA4=AAORA4=BBORA4=CCTHENGOSUB700
508 IFK=4THEN550
509 IFA5=EETHENGOSUB600
510 IFA1=EEORA2=EEORA3=EEORA4=EETHENGOSUB700
511 IFA5=AAORA5=BBORA5=CCORA5=DDTHENGOSUB700
550 PRINT"■";TAB(15);V;TAB(21);J;TAB(28);N;TAB(32)
551 J=0:N=0
552 GOTO25
600 J=(J+1)
601 IFJ=KTHENGOTO2000
603 RETURN
700 N=(N+1)
701 RETURN
1000 IFA=AAORA=BBORA=CCORA=DDTHEN15
1004 IFK=4THEN1010
1005 IFA=EETHEN15
1010 ONXGOTO1100,1200,1300,1400,1500
1100 AA=A:GOTO20
1200 BB=A:GOTO20
1300 CC=A:GOTO20
1400 DD=A:GOTO20
1500 EE=A:GOTO20
2000 PRINT:PRINT"■■";TAB(15);V;TAB(21);J;TAB(28);N;TAB(32)
2002 PRINT:PRINT:PRINT"HERZLICHEN GLUECKWUNSCH, DU HAST RICHTIGGERATEN !!!"
2004 PRINT:PRINT:PRINT"WUENSCHST DU EIN NEUES SPIEL? [J/N ":INPUTY$
2006 IFY$="J"THEN1
2008 PRINT"■":PRINT"VIELEN DANK FUER DAS SPIEL. AUF WIEDER- SEHEN.":END
```

Basic-Listing des Superhirn-Programms. Die unterstrichenen Zeichen werden auf dem PET 2001 invers dargestellt

Michael Bleistein

Z-80-Assembler in Basic

Im folgenden wird ein 2-Pass-Assembler für die CPU Z-80 beschrieben. Da die zugrundeliegende Programmsprache Basic nicht prozessorspezifisch ist, kann das Programm durch kleine Änderungen an jeden anderen CPU-Typ adaptiert werden. Der Assembler ist für den Einsatz auf Systemen mit Floppy Disk geschrieben, kann aber leicht auf weniger komfortable Speichermedien umgestellt werden.

Der beschriebene Z-80-Assembler unterscheidet sich von anderen Konzepten durch die Verwendung der Programmsprache Basic. Hierdurch ist es möglich, die Arbeitsweise transparent zu machen, und individuelle Änderungen oder Erweiterungen sind ohne Komplikationen möglich. Der gegenüber einem in Maschinensprache geschriebenen Programm erforderliche Mehrbedarf an Speicherplatz – es müssen sowohl der Basic-Interpreter als auch das Benutzerprogramm geladen werden – fällt bei den meisten Mikroprozessoren nicht ins Gewicht, da der Basic-Interpreter resident vorhanden ist. Allerdings kann nicht verschwiegen werden, daß die Programmlaufzeit dieses Assemblers nicht mit der von maschinenorientierten Programmen verglichen werden kann.

Arbeitsweise eines Assemblers

Der Assembler ist ein verbreitetes Hilfsmittel, um die Programmierung in Maschinensprache zu erleichtern. Er erleichtert das Programmieren durch zwei wesentliche Funktionen: Erstens „versteht" der Assembler die mnemonischen Befehle, d.h. er kann sie in den maschinenlesbaren Objectcode übersetzen. Zweitens ist es während des Programmierens möglich, bestimmten Adressen symbolische Namen zuzuordnen. Diese symbolischen Namen, die auch als Operanden in Befehlen auftreten können, werden als Labels bezeichnet. Somit entfällt das Berechnen von Sprungzielen, was die Fehlermöglichkeiten stark verringert.

Da die Umsetzung der symbolischen Adressen in absolute erst erfolgen kann, wenn alle symbolischen Adressen bekannt sind, sind zwei Durchgänge erforderlich. Aus dieser Programmstruktur leitet sich die Bezeichnung 2-Pass-Assembler ab. Eine schematische Darstellung der Arbeistweise gibt *Bild 1*.

Zusätzlich zu den CPU-spezifischen mnemonischen Abkürzungen sind in einem Assembler noch sogenannte Pseudoanweisungen definiert. Diese Anweisungen dienen dazu, den Assemblierungsgang zu steuern. Es können die Startadresse vorgegeben werden und Datenbereiche mit Konstanten belegt oder reserviert werden. Eine Aufstellung der in diesem Programm definierten Pseudoanweisungen zeigt *Tabelle 1*.

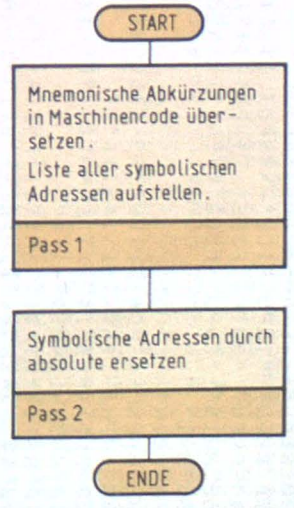
Tabelle 1: Pseudo-Anweisungen des Assemblers

ORG HHHH	Lädt den internen Programmzähler mit HHHH
DB HH	Definiert ein Datenbyte mit dem Wert HH
DW HHHH	Definiert ein Wort mit dem Wert HHHH
DS DD	Reserviert DD Bytes (DD Dezimalzahl)
END	Markiert Programmende
EQU HHHH	Ordnet dem Label vor EQU die Adresse HHHH zu

Programmbeschreibung

Bevor auf den logischen Aufbau des Programms näher eingegangen wird, müssen drei Unterschiede der zugrundeliegenden Basic-Implementierung von Digital Equipment gegenüber verbreiteten Mikroprozessorversionen erläutert werden.

Mehrfachanweisungen innerhalb einer Zeile werden nicht durch einen Doppelpunkt, sondern durch einen Schrägstrich getrennt. Der Befehl SEG$(X$,A,B) bildet aus dem String X$ den Teilstring, der sämtliche Elemente von A bis B umfaßt. In erweiterten Basic-Dialekten entspricht diesem Befehl die Anweisung MID$. Falls dieser Befehl nicht vorhanden ist, empfiehlt es sich, eine Benutzerfunktion mit entsprechendem Verhalten zu definieren, die anstelle des Befehls aufgerufen wird.



◄ Bild 1. Arbeitsweise eines 2-Paß-Assemblers

```
10 X$="FUNKSCHAU"
20 A=POS(X$,"U",3)
30 B$=SEG$(X$,1,4)
40 PRINT A,B$
50 END
RUN
DEMO     18-JUN-79   BASIC V01-05
 9              FUNK
```

Bild 2. Beispielprogramm zur Wirkungsweise der Anweisungen SEG$ und POS

```
10 REM ******************************************************
20 REM **********       Z-80 ASSEMBLER V 1.0       **********
30 REM ******************************************************
40 REM
50 REM
60 REM MICHAEL BLEISTEIN    28.03.1979
70 REM
80 REM *** INITIALISIERUNG DES PROGRAMMS
90 DIM A(3),A$(4),P(5)
100 S$="                                                    "
110 Z$="0000000000000000000000000000000000000000000000000000
    000000000000"
120 DEF FNS(L9)=SEG$(S$,1,L9)
130 DEF FNZ(L9)=SEG$(Z$,1,L9)
140 OPEN "DATA.ASS" AS FILE #7
150 DIM C$(700)
160 I=1
170 IF END #7 THEN 200
180 INPUT #7:C$(I)\I=I+1
190 GO TO 170
200 CLOSE #7
210 REM
220 REM *** EINGABEROUTINE
230 PRINT "INPUTFILE";\INPUT I$\F$=I$&".ASS"
240 PRINT "OUTPUTFILE";\INPUT I$\E$=I$&".ASS"
250 PRINT "INPUT MODE (Y/N)";\INPUT I$\IF I$="N" THEN 400
260 N=1
270 PRINT
280 OPEN F$ FOR OUTPUT AS FILE #1\I=0
290 INPUT I$\I=I+1\I$=I$&";"
300 A(1)=POS(I$,";",1)\A(2)=POS(I$,";",A(1)+1)
310 A(3)=LEN(I$)-1
320 A$(1)=STR$(I)\A$(1)=FNZ(4-LEN(A$(1)))&A$(1)&" "
330 A$(2)=SEG$(I$,A(0)+1,A(1)-1)\A$(2)=A$(2)&FNS(7-LEN(A$(2)))
340 A$(3)=SEG$(I$,A(1)+1,A(2)-1)\A$(3)=A$(3)&FNS(18-LEN(A$(3)))
350 A$(4)=SEG$(I$,A(2)+1,A(3))\A$(4)=A$(4)&FNS(19-LEN(A$(4)))
360 PRINT #1:FNS(14)&A$(1)&A$(2)&A$(3)&A$(4)
370 IF SEG$(A$(3),1,3)<>"END" THEN 290
380 PRINT \PRINT \CLOSE #1
390 REM
400 REM *** EDIT MODE
410 PRINT "EDIT MODE (Y/N)";\INPUT I$\IF I$="N" THEN 890
420 PRINT \PRINT "I=INSERT;D=DELETE;E=EXIT;L=LIST"\PRINT \PRINT
430 INPUT I$
440 IF SEG$(I$,1,1)="I" THEN 630 \IF SEG$(I$,1,1)="D" THEN 470
450 IF SEG$(I$,1,1)="L" THEN 830 \IF SEG$(I$,1,1)="E" THEN 890
460 PRINT "ILL INP"\GO TO 430
470 S9=VAL(SEG$(I$,2,5))
480 OPEN F$ FOR INPUT AS FILE #1
490 OPEN "ZZZZZZ.ASS" FOR OUTPUT AS FILE #2
500 FOR I=1 TO S9-1
510 INPUT #1:I$\PRINT #2:I$\NEXT I
520 INPUT #1:I$
530 IF END #1 THEN 570
540 INPUT #1:I$\H$=STR$(VAL(SEG$(I$,15,18))-1)
550 I$=SEG$(I$,1,14)&FNZ(4-LEN(H$))&H$&SEG$(I$,19,LEN(I$))
560 PRINT #2:I$\GO TO 530
570 CLOSE #1\CLOSE #2
580 OPEN F$ FOR OUTPUT AS FILE #1
590 OPEN "ZZZZZZ.ASS" FOR INPUT AS FILE #2
600 IF END #2 THEN 620
610 INPUT #2:I$\PRINT #1:I$\GO TO 600
620 CLOSE \GO TO 430
630 I$=SEG$(I$,2,LEN(I$))&";"
640 A(0)=POS(I$,";",1)\A(1)=POS(I$,";",A(0)+1)\A(2)=POS(I$,";
    ",A(1)+1)
650 A(3)=LEN(I$)-1
660 A$(1)=SEG$(I$,1,A(0)-1)\A$(1)=FNZ(4-LEN(A$(1)))&A$(1)&" "
670 A$(2)=SEG$(I$,A(0)+1,A(1)-1)\A$(2)=A$(2)&FNS(7-LEN(A$(2)))
680 A$(3)=SEG$(I$,A(1)+1,A(2)-1)\A$(3)=A$(3)&FNS(18-LEN(A$(3)))
690 A$(4)=SEG$(I$,A(2)+1,A(3))\A$(4)=A$(4)&FNS(19-LEN(A$(4)))
700 OPEN F$ FOR INPUT AS FILE #1
710 OPEN "ZZZZZZ.ASS" FOR OUTPUT AS FILE #2
720 X=VAL(A$(1))-1
730 FOR I=1 TO X\INPUT #1:I$\PRINT #2:I$\NEXT I
740 I$=FNS(14)&A$(1)&A$(2)&A$(3)&A$(4)\PRINT #2:I$
750 IF END #1 THEN 790
760 INPUT #1:I$\H$=STR$(VAL(SEG$(I$,15,18))+1)
770 I$=SEG$(I$,1,14)&FNZ(4-LEN(H$))&H$&SEG$(I$,19,LEN(I$))
780 PRINT #2:I$\GO TO 750
790 CLOSE \OPEN "ZZZZZZ.ASS" AS FILE #2\OPEN F$ FOR OUTPUT
    AS FILE #1
800 IF END #2 THEN 820
810 INPUT #2:I$\PRINT #1:I$\GO TO 800
820 CLOSE \GO TO 430
830 PRINT \PRINT F$;"   Z80 ASSEMBLER     V 1.0 "\PRINT \PRINT
840 OPEN F$ AS FILE #1
850 IF END #1 THEN 870
860 INPUT #1:I$\PRINT I$\GO TO 850
870 CLOSE #1\PRINT \PRINT \GO TO 430
880 REM
890 REM *** ERSTER ASSEMBLIERUNGSDURCHGANG
900 OPEN "LABLIS.ASS" FOR OUTPUT AS FILE #2
910 OPEN F$ FOR INPUT AS FILE #1
920 OPEN "FEHLIS.ASS" FOR OUTPUT AS FILE #3
930 OPEN "ZZZZZZ.ASS" FOR OUTPUT AS FILE #4
940 REM EINTRAGEN EINES LABELS
950 IF END #1 THEN 1280
960 INPUT #1:I$\IF SEG$(I$,20,25)="      " THEN 1020
970 IF SEG$(I$,27,29)="EQU" THEN 1000
980 D0=C0\GOSUB 2150
990 PRINT #2:SEG$(I$,20,25)&"/"&FNZ(4-LEN(H$))&H$\GO TO 1020
1000 PRINT #2:SEG$(I$,20,25)&"/"&SEG$(I$,31,34)
1010 REM SUCHEN DES CODES IN CODELISTE
1020 B$=SEG$(I$,27,42)\R$=""\I=0\C1=C0
1030 P(0)=POS(B$,"#",1)\P(1)=POS(B$,"#",P(0)+1)
```

```
1040 P(2)=POS(B$,"#",P(1)+1)\P(3)=POS(B$,"#",P(2)+1)
1050 P(4)=POS(B$,"$",1)\P(5)=POS(B$,"$",P(4)+1)
1060 IF P(0)<P(4) THEN 1070 \GO TO 1080
1070 I=I+4\GOSUB 1990 \I=I-6\GOSUB 1920 \GO TO 1090
1080 GOSUB 1920 \I=I+2\GOSUB 1990 \I=I-4
1090 GOSUB 1920
1100 GOSUB 2260
1110 IF POS(O$,"HH",1)=0 THEN 1200
1120 A$=SEG$(O$,1,POS(O$,"HH",1)-1)&SEG$(R$,1,2)
1130 O$=A$&SEG$(O$,POS(O$,"HH",1)+2,LEN(O$))&SEG$(R$,3,LEN(R$))
1140 GO TO 1210
1150 IF LEN(R$)<>4 THEN 1200
1160 B1$=SEG$(O$,1,2)\IF B1$="01" THEN 1200 \IF B1$="11"
     THEN 1200
1170 IF B1$="21" THEN 1200 \IF B1$="31" THEN 1200
1180 B1$=SEG$(O$,1,4)\IF B1$="DD21" THEN 1200 \IF B1$="FD21"
     THEN 1200
1190 R$=SEG$(R$,3,4)&SEG$(R$,1,2)
1200 O$=O$&R$
1210 O$=O$&FNS(12-LEN(O$))
1220 IF E=0 THEN 1260
1230 D0=C1\GOSUB 2150
1240 PRINT #4:FNZ(4-LEN(H$))&H$&" "&O$&SEG$(I$,14,62)
1250 GO TO 950
1260 PRINT #3:"ILL CODE AT LINE "&SEG$(I$,15,18)
1270 PRINT #4:I$\GO TO 950
1280 REM KOPIEREN VON FILE #4 AUF FILE  #1
1290 CLOSE #1\CLOSE #4\OPEN E$ FOR OUTPUT AS FILE #1
1300 OPEN "ZZZZZZ.ASS" AS FILE #4
1310 IF END #4 THEN 1330
1320 INPUT #4:I$\PRINT #1:I$\GO TO 1310
1330 CLOSE #1\CLOSE #2\CLOSE #4
1340 REM
1350 REM *** ZWEITER ASSEMBLIERUNGSDURCHGANG
1360 OPEN E$ AS FILE #1
1370 OPEN "LABLIS.ASS" AS FILE #2
1380 OPEN "ZZZZZZ.ASS" FOR OUTPUT AS FILE #4
1390 IF END #1 THEN 1650
1400 INPUT #1:I$\IF POS(I$,"LL",1)=0 THEN 1640
1410 L$=""
1420 P(4)=POS(I$,"$",1)\P(5)=POS(I$,"$",P(4)+1)
1430 L$=SEG$(I$,P(4)+1,P(5)-1)
1440 L$=L$&FNS(6-LEN(L$))
1450 RESTORE #2
1460 IF END #2 THEN 1620
1470 INPUT #2:X$\IF L$="      " THEN 1640
1480 IF L$=SEG$(X$,1,6) THEN 1500
1490 GO TO 1460
1500 B1$=SEG$(I$,6,7)\IF B1$="18" THEN 1530 \IF B1$="10"
     THEN 1530
1510 IF B1$="20" THEN 1530 \IF B1$="28" THEN 1530
1520 IF B1$="30" THEN 1530 \IF B1$="38" THEN 1530 \GO TO 1590
1530 H$=SEG$(I$,1,4)\GOSUB 2060 \A1=D0
1540 H$=SEG$(X$,8,11)\GOSUB 2060 \A2=D0
1550 A=A2-A1-2\IF A<-127 THEN 1630 \IF A>128 THEN 1630
1560 IF A<0 THEN 1570 \D0=A\GOSUB 2150 \GO TO 1580
1570 D0=256+A\GOSUB 2150
1580 X$=SEG$(X$,1,7)&"   "&H$
1590 A$=SEG$(I$,1,POS(I$,"LL",1)-1)&SEG$(X$,10,11)&SEG$(X$,8,9)
1600 I$=A$&SEG$(I$,POS(I$,"LL",1)+4,LEN(I$))
1610 GO TO 1640
1620 PRINT #3:"LBL NFD AT LINE "&SEG$(I$,15,18)\GO TO 1640
1630 PRINT #3:"JR OOR AT LINE"&SEG$(I$,15,18)
1640 PRINT #4:I$\GO TO 1390
1650 CLOSE #1\CLOSE #4\CLOSE #3\OPEN E$ FOR OUTPUT AS FILE #1
1660 OPEN "ZZZZZZ.ASS" AS FILE #4
1670 IF END #4 THEN 1690
1680 INPUT #4:I$\PRINT #1:I$\GO TO 1670
1690 CLOSE
1700 REM
1710 REM *** AUSGABEROUTINE
1720 PRINT "ASSEMBLIERUNG BEENDET"
1730 OPEN E$ AS FILE #1\PRINT \PRINT
1740 PRINT \PRINT \PRINT F$;"              Z-80 ASSEMBLER V 1.0"
1750 PRINT \PRINT
1760 PRINT "ADR";TAB(5);"OBJ CODE";TAB(18);"STMT";TAB(23);
1770 PRINT "LABEL";TAB(30);"SOURCE CODE";TAB(48);"COMMENT
1780 PRINT
1790 IF END #1 THEN 1810
1800 INPUT #1:I$\PRINT I$\GO TO 1790
1810 CLOSE #1\OPEN "FEHLIS.ASS" AS FILE #3\PRINT \PRINT \PRINT
     "FEHLER"
1820 IF END #3 THEN 1840
1830 INPUT #3:I$\PRINT I$\GO TO 1820
1840 CLOSE
1850 OPEN "LABLIS.ASS" FOR INPUT AS FILE #1
1860 PRINT \PRINT \PRINT "LABEL LISTING"\PRINT
1870 IF END #1 THEN 1900
1880 INPUT #1:I$\PRINT SEG$(I$,1,6)&"   "&SEG$(I$,8,11)
1890 GO TO 1870
1900 CLOSE \END
1910 REM
1920 REM *** UP ELIMINIERT HEXADEZIMALKONSTANTEN
1930 IF P(I)=0 THEN 1960
1940 R$=R$&SEG$(B$,P(I)+1,P(I+1)-1)
1950 B$=SEG$(B$,1,P(I)-1)&SEG$(B$,P(I+1)+1,LEN(B$))
1960 I=I+2
1970 RETURN
1980 REM
1990 REM *** UP ELIMINIERT SYMBOLISCHE ADRESSEN
2000 IF P(I)=0 THEN 2030
2010 B$=SEG$(B$,1,P(I)-1)&SEG$(B$,P(I+1)+1,LEN(B$))
2020 R$=R$&"LL"
2030 I=I+2
2040 RETURN
2050 REM
```

```
2060 REM *** UP HEX>DEZ ; H$>D0
2070 D0=0
2080 FOR I=LEN(H$) TO 1 STEP -1
2090 T$=SEG$(H$,I,I)\V=ASC(T$)
2100 IF V<65 THEN 2120
2110 V=V-55\D0=D0+16^(LEN(H$)-I)*V\GO TO 2130
2120 V=V-48\D0=D0+16^(LEN(H$)-I)*V
2130 NEXT I\RETURN
2140 REM
2150 REM *** UP DEZ>HEX ; D0>H$
2160 T$=""\D9=D0
2170 H$=""
2180 D9=D9/16\D8=(D9-INT(D9))*16\D9=INT(D9)
2190 IF D8>9 THEN 2210
2200 T$=T$&STR$(D8)\GO TO 2220
2210 D8=D8+55\T$=T$&CHR$(D8)
2220 IF D9<>0 THEN 2180
2230 FOR I=LEN(T$) TO 1 STEP -1
2240 H$=H$&SEG$(T$,I,I)\NEXT I\RETURN
2250 REM
2260 REM *** UP SUCHT BEFEHL B$ IM FELD C$ ; KORRIGIERT PC
2270 FOR I=LEN(B$) TO 1 STEP -1
2280 A$=SEG$(B$,I,I)\IF A$<>" " THEN 2300
2290 B$=SEG$(B$,1,I-1)\NEXT I
2300 E=0
2310 IF SEG$(B$,1,3)<>"ORG" THEN 2340
2320 H$=SEG$(B$,5,8)\O$=""\GOSUB 2060 \C0=D0
2330 E=1\GO TO 2530
2340 IF SEG$(B$,1,2)<>"DS" THEN 2360
2350 C0=C0+VAL(SEG$(B$,4,LEN(B$)))\O$=""\GO TO 2510
2360 IF SEG$(B$,1,2)<>"DB" THEN 2380 \C0=C0+1\O$=SEG$
     (B$,4,5)
2370 GO TO 2510
2380 IF SEG$(B$,1,3)<>"END" THEN 2400 \C0=C0+1\O$=""
2390 GO TO 2510
2400 IF SEG$(B$,1,2)<>"DW" THEN 2420 \C0=C0+2\O$=SEG$
     (B$,4,7)
2410 GO TO 2510
2420 FOR I=1 TO 700\IF B$=SEG$(C$(I),1,POS(C$(I),"/",1)-1)
     THEN 2450
2430 NEXT I
2440 E=0\GO TO 2530
2450 O$=SEG$(C$(I),POS(C$(I),"/",1)+1,LEN(C$(I)))
2460 IF I>617 THEN 2500 \IF I>543 THEN 2490 \IF I>201
     THEN 2480
2470 C0=C0+1\GO TO 2510
2480 C0=C0+2\GO TO 2510
2490 C0=C0+3\GO TO 2510
2500 C0=C0+4
2510 E=1
2520 D0=C0\GOSUB 2150
2530 RETURN
2540 END
```

Bild 3. Basic-Listing des 2-Paß-Assemblers

Mit der Anweisung POS(X$,Y$,A) bestimmt man im DEC-Basic die Position des Strings Y$ in X$, wobei erst Elemente ab der A-ten Stelle berücksichtigt werden. In einigen Basic-Dialekten findet sich dieser Befehl als Anweisung INSTR. Ist sie nicht vorhanden, empfiehlt sich wieder das obige Verfahren. *Bild 2* zeigt ein kurzes Programm, welches die Arbeitsweise beider Befehle verdeutlicht.

Das Programm selbst ist in sechs Teile und fünf Unterprogramme gegliedert. *Bild 3* zeigt das komplette Listing. Die Zeilen 80 bis 200 bilden den Programmstart. Es werden die benötigten Felder dimensioniert, die Benutzerfunktionen definiert und die Codetabelle eingelesen. Auf eine Vorgabe der Tabelle durch DATA-Anweisungen wurde verzichtet, um den Speicherbedarf nicht zu erhöhen. *(Schluß folgt)*

Meine Meinung...

Sollten Sie einen TRS-80 besitzen, so dürften Sie ein relativ glücklicher Mensch sein, wenn Sie sich hübsch an das Bedienungsbuch halten und Ihre Programme in BASIC schreiben. Die Situation ändert sich aber schlagartig, wenn Sie sich der Tatsache erinnern, daß im TRS-80-Innern eine Z-80-CPU sitzt, die sich äußerst effizient in Maschinensprache programmieren läßt. Über die Vorteile von Maschinensprache wurde bereits einiges geschrieben, und tatsächlich gibt es sogar Hex-Monitorprogramme oder gar Assembler für den TRS-80 käuflich zu erwerben.

Allerdings: Solche Monitoren sind absolut nichts wert, solange man nicht weiß, welche Speicherplätze überhaupt frei sind und nicht vom BASIC belegt werden, wo die Adressen der einzelnen ICs im TRS-80 liegen und welche ROM-Unterprogramme für den Benutzer abrufbar sind. Es ist mir absolut rätselhaft, warum Tandy den Mantel des Schweigens über solch elementare Dinge hält.

Die Computerhersteller müßten sich schließlich doch darüber im klaren sein, daß das Fehlen einer guten ROM-Dokumentation zur Folge haben muß, daß eine Reihe unautorisierter, u. U. fehlerhafter Monitorprogramm-Beschreibungen zu kursieren beginnen.

Ein Gegenbeispiel ist der AIM-65 von Rockwell: Hier wird ein kleines Büchlein mitgeliefert, das das komplette Assembler-Listing des immerhin 8 KByte umfassenden Monitors mit verständlichen Kommentaren enthält; ähnlich ist es beim KIM-1, beim SYM-1, beim Eurocom-1 und bei vielen anderen Einplatinen-Computern.

Dagegen ist Commodore ähnlich zugeknöpft wie Tandy und mit Informationen über die Innereien der unterschiedlichen PET-Versionen fast genauso zurückhaltend. Eigentlich schade – denn erst bei genauer Kenntnis des Computer-Innenlebens lassen sich wirklich effiziente Programme schreiben. Ob die Hersteller das nicht wissen? Ich war immer der Meinung, daß sie versuchen würden, möglichst viele Computer zu verkaufen – und eine gute Dokumentation ist doch d a s Verkaufsargument...

Herwig Feichtinger

Zwischen Taschenrechner und Computer

HP-41-Testbericht

Einige Zeit war es um die Rechnerentwicklung bei Hewlett-Packard recht still geworden. Seit Texas Instruments den bisher wohl komplexesten programmierbaren Taschenrechner, den TI-59, vorstellte, war sogar ein wenig der Eindruck entstanden, HP wollte oder könnte nicht mehr mithalten. Mit dem neuen HP-41 ist man aber gründlich eines Besseren belehrt worden.

Software-kompatibel zum HP-67

Die Entwickler des HP-41 (*Bild 1*) gaben sich alle Mühe, dafür zu sorgen, daß jene Anwender, die schon einmal mit einem HP-Rechner gearbeitet haben, nicht völlig umdenken müssen. So ist der Befehlsvorrat der Typen HP-67 und HP-97 sozusagen eine „Teilmenge" der HP-41-Befehle, und sogar die Magnetkarten beider Familien sind miteinander austauschbar, solange nicht die zusätzlichen HP-41-Befehle verwendet werden. Mit anderen Worten: Jedes HP-67- oder HP-97-Programm läuft ohne weiteres auf dem neuen Rechnermodell. Zur Kompatibilität gehört auch die Beibehaltung der Umgekehrten Polnischen Notation UPN, die eine sehr rechnernahe und klammerfreie Eingabe algebraischer Zusammenhänge darstellt, und in Verbindung damit die vier Stack-Register X, Y, Z und T. Dazu kommt noch das bereits vom HP-25 bekannte Last-X-Register.

Gegenüber dem HP-25 ist der HP-41 (ebenso wie die 67/97-Familie) um rund 30 % langsamer in der Ausführung von Programmen. Damit besitzt auch der neue Rechner den bisher allen „Programmierbaren" anhaftenden Nachteil der gegenüber BASIC-Computern relativ geringen Verarbeitungsgeschwindigkeit.

Alphanumerische Ein- und Ausgabe

Der wesentlichste Fortschritt gegenüber den Rechnern HP-67 und HP-97 ist wohl in der Möglichkeit zu sehen, außer Ziffern auch Buchstaben eintippen und auf dem zwölfstelligen 16-Segment-LCD-Display erscheinen lassen zu können. Bei Programmen, die bestimmte Benutzereingaben erfordern, kann man im Display also mittels eines Prompt-Befehls einen entsprechenden Text erscheinen lassen, z.B. „R=?". Allerdings ist hier bereits eine gewisse Einschränkung erwähnenswert; da die „Alpha-Strings" keine Ziffern enthalten können, ist es nicht ohne weiteres möglich, z.B. „R1=?" anzuzeigen.

Bei der Ausgabe von Rechenergebnissen kann dagegen das für die Textspeicherung vorgesehene Alpha-Register mit einem beliebigen anderen Datenregister kombiniert angezeigt werden, z.B. in der Form „R=1.5000".

Die alphanumerische Eingabe wird auch für bestimmte Befehle benötigt, die nicht auf die Tasten aufgedruckt sind und daher als Folge mehrerer Buchstaben eingetippt werden müssen. Die Umschaltung in den Alpha-Eingabemodus geschieht mit einem Druck auf die Alpha-Taste; ein nochmaliger Druck schaltet in den numerischen Modus zurück. Um die Funktion CLP auszuführen sind also z.B. sechs Tastendrücke erforderlich: XEQ, ALPHA, C, L, P, ALPHA. Dies kann allerdings reduziert werden, indem man solche „versteckten" Funktionen bestimmten Tasten mit der Assign-Taste zuweist. In dem mit der User-Taste erreichbaren Sondermodus sind nämlich alle Rechnertasten vom Anwender selbst neu definierbar. Ebenso ist es möglich, in diesm User-Modus den gespeicherten selbstgeschriebenen Programmen bestimmte Tasten zuzuordnen, um deren Abruf zu vereinfachen.

Selbstgeschriebene Programme können mit Alpha-Labels versehen werden, z.B. „LC" oder ähnliches. Die Funktion CAT1 kann dann die Namen aller gespeicherten Programme auflisten. Auch können die Programme selbst andere Programme mit alphanumerischen Labels aufrufen und so als Unterprogramm verwenden. *Bild 2* und die *Tabelle* zeigen ein HP-41-Programmbeispiel.

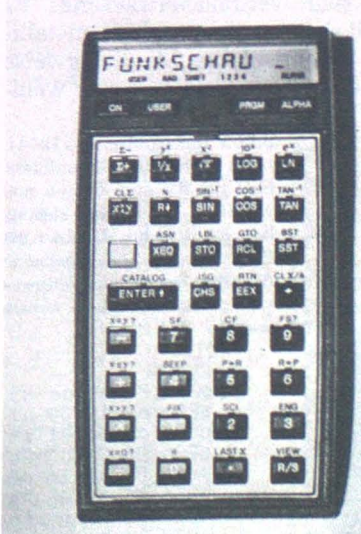
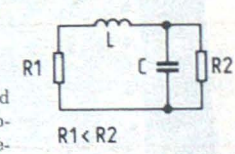


Bild 1. Der HP-41 gestattet neben der Zifferndarstellung auch die Anzeige von Buchstaben und Zeichen im LCD-Display



$R1 < R2$

$$L = \frac{\sqrt{R1 \cdot R2 - R1^2}}{2 \pi f}$$

$$C = \frac{L}{R1 \cdot R2}$$

Bild 2. Schaltbild und Formeln für das Programmbeispiel. Bei gegebenen R1 und R2 können für eine bestimmte Frequenz f die Werte von L und C errechnet werden.

CMOS-Speicher

Der gesamte RAM-Speicherbereich des Mikrocomputers im HP-41 ist in CMOS-Technik ausgeführt und wird auch dann nicht von der Batterie getrennt, wenn der Rechner ausgeschaltet wird. Während des Batteriewechsels kann der HP-41 dank eines Puffer-Elkos den Speicherinhalt noch mehrere Minuten behalten. Der CMOS-Speicher umfaßt alle Stack- und Datenregister sowie alle Programmspeicher und kann per SIZE-Befehl zwischen Daten- und Programmspeicher frei verteilt werden,

Beispiel: Errechnen von L und C für eine Anpaßschaltung

```
01 LBL"LC          17 "FREQ = ?
02 "R EINS = ?     18 PROMPT
03 PROMPT          19 /
04 STO 01          20 ENTER↑
05 "R ZWEI = ?     21 ENTER↑
06 PROMPT          22 RCL 01
07 STO 02          23 /
08 x               24 RCL 02
09 RCL 01          25 /
10 x↑2             26 "C =
11 –               27 ARCL x
12 SQRT            28 AVIEW
13 2               29 STOP
14 /               30 "L =
15 PI              31 ARCL Y
16 /               32 AVIEW
                   END
```

Programmlauf:
XEQ LC

```
R EINS = ?      FREQ = ?
50              150 E 6 R/S
R ZWEI = ?      C = 8,488 E – 12 R/S
250 R/S         L = 106,1 E – 9
```

(Die Benutzereingaben sind **halbfett** *gedruckt).*

wie dies bereits vom TI-59 her bekannt ist. Das Löschen des gesamten Speichers ist übrigens möglich, indem man schnell abwechselnd die ON- und die CLX-Taste drückt, bis auf dem Display „MEMORY LOST" erscheint.

Der Speicher ist (in der Grundausstattung) in 64 Registern organisiert. Jedes dieser Register kann wahlweise sieben Buchstaben, eine Fließkomma-Zahl oder sieben Programmschritte enthalten. Für das Einfügen und Löschen von Programmschritten sind ebenfalls Befehle vorhanden, wobei keine Korrektur von Sprungadressen erforderlich wird, da ausschließlich die symbolische Label-Adressierung verwendet werden kann.

Noch ein Wort zum Stromverbrauch: Ein Satz Batterien (NiCd-Akkus wären absolut sinnlos, da ihre Selbstentladung höher ist als der Rechnerverbrauch) reicht nach Angaben von HP rund 10 Monate. Aus diesem Grunde wird auch kein Ladegerät für den Rechner mitgeliefert. *Bild 3* zeigt den Innenaufbau des HP-41.

Peripherie

Eine äußerst praktische Sache ist beim HP-41, daß sich an vier „Bus-Steckleisten" am oberen Ende externe Geräte aufstecken lassen: Ein HP-67-kompatibler Magnetkartenleser, ein Thermodrucker und zusätzliche RAM- und ROM-Speichermoduln. Geplant ist auch ein „Bar Code Reader", also ein Lesegerät für den auf Verpackungen oft zu findenden Streifencode für Warennummer und Preis. Die mechanische Ausführung der Bus-Stecker ist leider nicht sehr vertrauenerweckend: Es handelt sich um eine gefaltete, metallisierte Kunststoffolie. Möglicherweise treten durch ihre Elastizität aber weniger Probleme als bei starren Verbindern auf.

Nur noch ein kleiner Schritt zum BASIC-Computer

Die Flexibilität des HP-41 reicht, wenn man einmal von der Rechengeschwindigkeit absieht, nahe an die eines Personal Computers heran, obwohl das Gerät nicht größer als vergleichbare Taschenrechner ist. Allerdings muß festgestellt werden, daß die Bedienung eines BASIC-Computers eher einfacher und übersichtlicher ist. Es bleibt also abzuwarten, wann alle Befehle nur noch alphanumerisch eingegeben werden, um ein langes Suchen nach selten benötigten Funktionstasten zu vermeiden.

Die Rechnerentwicklung ist also auch mit dem HP-41 noch keinesfalls abgeschlossen; es bleibt der Wunsch nach einer leichteren Bedienbarkeit, während die Zahl der zur Verfügung stehenden mathematischen Funktionen heute völlig ausreicht und die meisten BASIC-Interpreter bei weitem übertrifft. Fe.

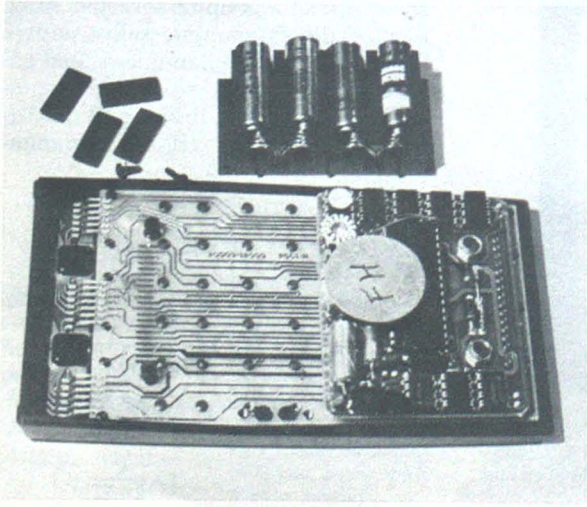


Bild 3. Innenaufbau des HP-41. Links oben die Gummifüsse, über die die zum Öffnen notwendigen Schrauben zugänglich sind, rechts daneben der Batteriekasten. Die Scheibe in der Mitte der Rechnerplatine ist ein piezoelektrischer Schallwandler

KIM spielt Lotto

Das im *Bild* als Hex-Dump abgedruckte 6502-Programm für den Mikrocomputer KIM-1 verwirklicht die „Ziehung" von Zufallszahlen nach dem Zahlenlotto-System, d.h. es darf keine Wertewiederholung auftreten. Die Startadresse ist 0200; die einzelnen Ziehungen werden mit der Taste 0 ausgelöst. Solange der Tastendruck anhält, durchläuft das Programm in schneller Folge und für den Benutzer unsichtbar die Zahlen 1...49. Beim Loslassen der Null-Taste prüft der Mikrocomputer, ob die nun ermittelte Zufallszahl schon einmal gezogen wurde; wenn ja, wird die Lottozahl dezimal um 1 erhöht, und es erfolgt noch einmal ein Test, ob die Zahl bereits gezogen wurde.

Auswahlwette-Spieler können das Programm auch verwenden; dazu muß nur die Adresse 020C mit 45 geladen werden. In der Zero-Page werden außer dem Display-Bereich (F9...FB) noch die Zellen 0...8 verwendet. Viel Glück beim Ziehen! *Raimund Pourvoyeur*

```
0200 F8 A9 00 85 06 A9 99 85 08 A5 08 C9 49 D0 02 A9
0210 00 18 69 01 85 08 20 6A 1F C9 00 F0 3D A5 07 C9
0220 00 D0 21 A5 08 C5 00 F0 E0 C5 01 F0 DC C5 02 F0
0230 D8 C5 03 F0 D4 C5 04 F0 D0 A5 06 8D 41 02 A5 08
0240 85 02 E6 06 A9 FF 85 07 A5 06 85 FA A9 00 85 FB
0250 A5 08 85 F9 20 1F 1F 4C 16 02 A9 00 85 07 85 FB
0260 85 FA 85 F9 20 1F 1F A5 06 C9 06 D0 09 A9 00 A2
0270 06 95 00 CA 10 FB 4C 09 02
```

Ob der KIM-1 beim Lottospielen mehr Glück hat als sein Operator, bleibt abzuwarten – jedenfalls erspart einem dieses Programm das Grübeln über die „richtigen" Zahlen...

HP-25 berechnet Frequenzgang

Programmierbare Taschenrechner besitzen meist nur eine eingeschränkte Zahl von Datenspeichern; beim HP-25 sind es acht. Trotzdem lassen sich damit mehr als acht Zahlenwerte speichern, wie das folgende Programmierbeispiel zeigt.

Ziel des Programms ist es, den Frequenzgang einer Schaltung zu errechnen, die maximal drei Tief- und/oder Hochpässe enthält, die voneinander entkoppelt sind. *Bild 1* macht die mathematischen Voraussetzungen für jeden Hoch- und Tiefpaß deutlich. Genaue Ergebnisse sind auch zu erwarten, wenn die drei Pässe nicht voneinander entkoppelt sind, sondern lediglich aufgrund ihrer Dimensionierung die Bedingungen von Bild 1 erfüllen, d.h. wenn der dem Generator nächste Pass niederohmig ausgelegt ist und die folgenden z.B. um den Faktor 10 hochohmiger werden.

Natürlich ist es nur mit einem Trick möglich, in den wenigen Datenspeichern des HP-25 alle benötigten Werte zu speichern. Dieser Trick besteht darin, die INT- und FRAC-Funktionen zum Zerlegen eines Speichers zu verwenden. Der Teil vor dem Dezimalpunkt speichert den Wert für R, der Nachkommateil denjenigen für C. Wie aus *Bild 2* hervorgeht, enthält das Vorzeichen der sich ergebenden Gesamtzahl außerdem die Aussage, ob es sich um einen Hoch- oder Tiefpaß handelt.

Bild 3 zeigt das Programm; es belegt alle 49 Programmschritte des HP-25. Nachdem es geladen ist, gibt man die Werte der drei Hoch- oder Tiefpässe in die Register 1...3 ein. Dann tippt man die zu untersuchende Frequenz in Hz ein und drückt R/S. Nacheinander erscheinen kurz die Einzeldämpfungen im Display, bis das Programm anhält und die Gesamtdämpfung bei der jeweiligen Frequenz in dB anzeigt. Nun kann man eine neue Frequenz eingeben, wieder R/S drücken usw.

Für alle, die gern anderer Leute Programme zerpflücken, hier noch die Belegung der übrigen Datenspeicher: In R 0 wird die eingegebene Frequenz, in R 4 die jeweilige Grenzfrequenz (neg. bei Tiefpaß) gespeichert; R 5 ist nicht belegt, R 6 gibt an, welcher Hoch- oder Tiefpaß gerade berechnet wird (0...2), und in R 7 werden die Dämpfungswerte in dB aufsummiert. Fe.

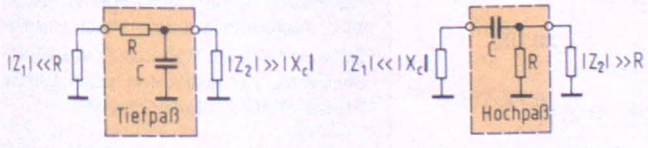


Bild 1. Die drei in der zu berechnenden Gesamtschaltung vorkommenden Elementar-Filter können ein R-C-Hochpaß oder ein R-C-Tiefpaß sein

Bild 3. ▶
Das Programm belegt alle 49 Schritte des HP-25. Will man die Dämpfung von weniger als drei Elementarfiltern errechnen, so kann man in die nicht benutzten Datenspeicher (z. B. R 3) den Wert – 1,000001 schreiben

01	STO 0	26	RCL 0
02	0	27	x
03	STO 6	28	g Pi
04	STO 7	29	x
05	RCL 6	30	2
06	g x=0	31	x
07	GTO17	32	g1/x
08	1	33	EEX
09	f x=y	34	6
10	GTO19	35	x
11	CLX	36	STO 4
12	2	37	g x≥0
13	f x=y	38	x⟷y
14	GTO21	39	÷
15	RCL 7	40	1
16	GTO00	41	STO+6
17	RCL 1	42	g →P
18	GTO22	43	f log
19	RCL 2	44	2
20	GTO22	45	0
21	RCL 3	46	x
22	ENTER	47	STO+7
23	f INT	48	PAUSE
24	x⟷y	49	GTO05
25	gFRAC		

◀ Bild 2.
In den HP-25-Registern R 1...3 werden sechs Bauelementewerte gespeichert: R 1 enthält die Werte des ersten Elementarfilters, R 2 die des zweiten und R 3 die des dritten

µC-Nachrichten

Die Berliner Firma Runow hat einen **Personal Computer** namens PAC (Personal Allround Computer) herausgebracht, der in Basic programmierbar ist, eine große Normtastatur enthält und über zwei BASF-Minifloppy-Laufwerke verfügt. Er arbeitet mit der CPU 6502 und kostet mit 32 KByte RAM etwa 7000 DM.

□

Erika Szabó in Puchheim/München entwickelte einen **Basic-Generator** für den TRS-80, der symbolische Adressierung statt Zeilennummern gestattet und den Programmierer bei der Planung, Codierung und Dokumentation von Basic-Programmen unterstützt. Durch die strukturierte Programmierung gewinnt man bis zu 30 % Zeit. Der Generator ist auf Kassette lieferbar und belegt 4 KByte.

□

SGS-Ates hat für das Mikrocomputer-System CLZ-80 einen **Basic-Interpreter** herausgebracht, der 8 KByte belegt und in vier 2716-EPROMs gespeichert ist. Befehle und Syntax entsprechen denen der heute üblichen Personal Computer.

□

Ein **Sprachsynthese-Modul** hat A.E.M. in München herausgebracht. Es wird über eine 6-bit-ASCII-Leitung parallel angesteuert und beherrscht 24 Wörter als „Rechner-Vokabular", 64 Wörter Standard-Vokabular und 64 ASCII-Wörter. Der Synthesizer ist für die Sprachen Englisch, Deutsch und Französisch erhältlich. Ein besonderer Vorteil ist, daß keine speziellen Steuerprogramme erforderlich sind.

□

Ein Report von Mackintosh prophezeit für 1983 einen Marktanteil von 23 % für **16-bit-Mikroprozessoren** (heute: 6 %). Für die nächsten drei bis fünf Jahre sei der Typ 8086 die dominierende 16-bit-CPU. Es wird auch darauf hingewiesen, daß kein einziger Hersteller bisher konsequenterweise auch 16-bit-Speicherbausteine entwickelte. Auch in einigen Jahren werden die 8-bit-CPUs also noch die Überhand behalten.

□

Sylvania begann mit dem Vertrieb des von Mattel gebauten Heimcomputers **„Intellivision"**, der einen 16-bit-Prozessor von General Instruments verwendet. Nach Magnavox ist Sylvania der zweite amerikanische Fernsehgerätehersteller, der in das Heimcomputer-Geschäft einsteigt.

Computer-Clubs

Eine Reihe von Clubs hat sich inzwischen in den deutschsprachigen Ländern etabliert, deren Zweck es ist, Programme und Erfahrungen auszutauschen. Eine Anzahl von ihnen ist mehr oder weniger firmengebunden – damit befaßt sich unsere Glosse.

Name des Clubs	Anschrift	Mitglieder	Systeme	Jahresbeitrag
TRS-80-Anwender-Club	W. u. G. Luce Im Ortfeld 12 4300 Essen 14	140	TRS-80	48 DM
TI-58/59-Software-Club	P. G. Poloczek Kalbacher Hauptstr. 71 6000 Frankfurt 56	80	TI-58/59	42 DM
Programm-Tausch-Club	D. Severitt Gutenbergstr. 20 6052 Mühlheim	50	PET	30 DM
Siemens-Gruppe	Postfach 103 8000 München 1	250	8080 u. a.	–
Small Business Computer Club	Bruchstr. 50 CH-6003 Luzern	35	Basic	30 Fr.
Taschenrechner-Anwender-Club	H. Schnepf Buchenweg 24 5000 Köln 40	1200	Taschenrechner u. PET	48 DM
7768-Benutzergruppe	P. Bendall Flottmoorring 67, 2358 Kaltenkirchen	480	Newbear 7768	–
Hamburger Computerclub	Günter Giese Kampstr. 34 2085 Quickborn	30	alle	60 DM
Personal Computer Club	Augustenstr. 77 8000 München 40	15	TRS-80, PET	20 DM
HP User's Club	Rue du Bois-du-Lan, CH-1217 Genf/Merin-2	10000	HP-Rechner	40 Fr.
Heath User's Group	Postfach 102060 6072 Dreieich	?	H-8, H-11, WH-89	48 DM
EDV-Benutzerverband	Ottokar Schöse Postfach 29 8473 Pfreimd	?	alle	360 DM
Computer-Club Schweiz	A. Stolz Lindenstr. 30 CH-9204 Andwil	40	alle	–
AMMS	Bernd Walter Torstr. 27 7000 Stuttgart	62	alle	30/60 DM
Apple User Group Europe	Postfach 4068 4320 Hattingen 13	?	Apple II	50 DM
AG-Mikrocomputer Frankfurt	K. D. Friedrich Bernadottestr. 3 6 Frankfurt 50	ca. 20	alle	–
Malmö Amatör Datorister	L. E. Lind P. B. 46 S-232 00 Arlöv Schweden		8080/Z-80	100 SKr

Computer-Clubs und andere Lustbarkeiten

Was tun drei Deutsche, die sich treffen? Sie gründen einen Verein. Was halten Sie davon, wenn die XY-Autowerke einen Club gründen zur Verbreitung des Wissens um das Auto, die Kernkraftwerke-Hersteller einen Verein „Freunde der Kernenergie" zur Verbreitung des Wissens um die Kernkraft?

Hingegen ist offenbar gar nichts dabei, wenn eine Computer-Verkaufs-GmbH & Co. KG einen Verein zur Verbreitung des Wissens über Computer gründet. Dieser Verein gibt eine Clubzeitschrift heraus, in der die Computer-Verkaufs-GmbH & Co. KG kräftig in den Vordergrund gestellt wird. Natürlich erhalten Vereinsmitglieder einen kleinen Rabatt auf die Produkte der Firma; wer weiß denn schon, daß die Preise ohnehin überhöht sind?

Mit der Clubzeitschrift ist das so eine Sache: Daran kann man schlecht verdienen und Arbeit macht sie auch noch. Also werden Clubmitglieder angeheuert, Artikel zu schreiben – lang und breit. Beim Geld wird es schwieriger: Der Verein muß ja gemeinnützig sein, folglich wird es aus bezahlter Arbeit für die Zeitschrift nichts. Und auch das Geld, das die Anzeigen darin bringen, geht ja in die Vereinskasse.

Auch dieses Problem wird gemeistert: Flugs gründet man eine Vereins-Zeitschrift-GmbH & Co. KG. Geld ist Geld – Adressen bringen übrigens auch Geld...

F. D., Stuttgart

Michael Konz

6502 simuliert 8080

Oft findet der Computeranwender, der vielleicht einen AIM-65, einen KIM-1 oder einen SYM-1, also ein 6502-System besitzt, ein Programm, das ihn zwar brennend interessiert, das er aber auf seinem Computer nicht laufen lassen kann, weil es für einen anderen Prozessor geschrieben ist, etwa den 8080. Solange dieses Programm keine Echtzeitverarbeitung betreibt, kann dem mit einem Simulator, wie er im folgenden geschildert wird, abgeholfen werden.

Ein Simulator ist praktisch ein Interpreter; nur wird dabei keine höhere Programmiersprache wie BASIC in einen Maschinencode übersetzt, sondern ein Maschinencode in einen anderen [1,2]. Daß die Ausführungszeiten des simulierten Prozessors dabei gewaltig steigen, ist selbstverständlich, kann aber oft hingenommen werden.

Dieser Simulator (*Bild 1*) bildet ein 8080-System softwaremäßig auf einem 6502-Mikrocomputer nach. Die Arbeitsregister des Pseudo-8080 befinden sich dabei in der Zero-Page, und zwar an folgenden Adressen:

Register	*Adresse*
Programmzähler PC	AA,AB
Stackpointer SP	A6, A7
Statusregister F	A8
Akku A	A9
H, L	A5, A4
D, E	A3, A2
B, C	A1, A0

Bei Stackoperationen, z. B. Unterprogramm-Aufrufen, darf man keinesfalls vergessen, den Stackpointer vorher zu laden, z. B. die Zelle A6 mit dem Wert D0 und A7 mit 01, so daß der Pseudo-8080 wie der 6502 Page 1 als Stack verwendet.

Außerdem werden die Plätze 00...60 der Zero-Page sowie AD, AE, AF, F0 und F1 benötigt. Das Programm selbst belegt die Speicherplätze 0200...0615,

```
0000 E9 0A 1A 02 025C 00 B1 02 BA 031C 66 A9 4C 10 03DC 91 A6 60 B1 049C 29 C7 85 A8 055C 20 01 06 68
0004 12 E3 F9 EB 0260 02 BF 02 C4 0320 03 A5 A8 4A 03E0 A6 48 A2 06 04A0 A5 A9 A2 00 0560 4C 40 05 20
0008 27 2F 37 3F 0264 02 C9 02 CF 0324 4C 0E 03 A5 03E4 20 38 03 B1 04A4 4A 90 01 E8 0564 68 03 A9 06
000C 07 0F 17 1F 0268 02 DE 02 E7 0328 A8 4C 1B 03 03E8 A6 48 20 38 04A8 C8 C0 08 90 0568 85 F1 6C F0
0010 00 C9 76 D3 026C 02 A0 05 F6 032C B1 AA 29 30 03EC 03 20 BD 03 04AC F7 8A 4A B0 056C 00 EA EA EA
0014 DB C3 CD 2A 0270 02 FD 02 04 0330 4A 4A 4A AA 03F0 68 95 A1 68 04B0 06 A5 A8 09 0570 B1 AA 29 38
0018 3A 22 32 C0 0274 03 0B 03 19 0334 60 20 2C 03 03F4 95 A0 60 20 04B4 06 D0 04 A5 0574 4A 4A 4A EA
001C C0 CF CF C7 0278 03 21 03 27 0338 F6 A0 D0 02 03F8 2C 03 B5 A0 04B8 A8 29 D3 85 0578 4A EA AA B5
0020 C7 CF CF CF 027C 03 0A 03 4E 033C F6 A1 60 20 03FC 65 A4 85 A4 04BC A8 A5 A9 08 057C 5D 24 A8 F0
0024 C7 C7 C7 C7 0280 03 62 03 63 0340 2C 03 38 B5 0400 B5 A1 65 A5 04C0 68 29 82 85 0580 01 60 90 02
0028 C7 CF 40 80 0284 05 63 05 76 0344 A0 E9 01 95 0404 85 A5 4C 10 04C4 AD 0A 0A 0A 0584 18 60 38 60
002C C1 C5 04 05 0288 03 80 03 93 0348 A0 B0 02 D6 0408 03 20 68 03 04C8 0A 0A 05 AD 0588 20 70 05 90
0030 03 0B 09 C0 028C 03 A0 03 A8 034C A1 60 B1 A6 040C 20 2C 03 A5 04CC 85 AD A5 A8 058C FA 4C 76 03
0034 06 C6 C2 C4 0290 03 B5 03 5D 0350 85 AA A2 06 0410 F0 95 A0 A5 04D0 29 17 05 AD 0590 20 70 05 90
0038 01 13 15 1B 0294 04 53 04 DF 0354 20 38 03 B1 0414 F1 95 A1 60 04D4 85 A8 60 A5 0594 F2 4C 80 03
003C 25 27 2A 01 0298 03 C7 03 2F 0358 A6 85 AB A9 0418 B1 AA 29 38 04D8 A8 4A 4C 77 0598 20 70 05 90
0040 02 03 01 02 029C 05 4F 05 35 035C 03 85 AF 4C 041C 4A 4A 4A C9 04DC 04 38 4C F7 059C EA 4C 4E 03
0044 03 76 04 D7 02A0 03 3F 03 F7 0360 38 03 A2 FF 0420 06 F0 09 AA 04E0 05 48 E5 AD 05A0 A5 A8 29 10
0048 04 DD 04 FE 02A4 03 98 05 6A 0364 9A 4C 82 E1 0424 B5 55 85 F0 04E4 85 A9 20 10 05A4 F0 22 24 AE
004C 04 06 05 12 02A8 04 3C 04 88 0368 C8 B1 AA AA 0428 84 F1 EA 60 04E8 03 20 04 03 05A8 10 17 A5 A9
0050 05 1E 05 25 02AC 05 90 05 09 036C C8 B1 AA 86 042C A5 A5 85 F1 04EC 68 28 29 0F 05AC 38 E9 06 85
0054 05 A1 A0 A3 02B0 04 A5 A4 85 0370 F0 85 F1 A0 0430 A5 A4 85 F0 04F0 48 A5 AD 29 05B0 09 A5 A8 4A
0058 A2 A5 A4 00 02B4 AA A5 A5 4C 0374 00 60 20 68 0434 60 B1 AA 29 04F4 0F 85 AD 68 05B4 90 06 A5 A9
005C A9 40 01 04 02B8 7B 03 B1 A0 0378 03 86 AA 85 0438 07 4C 1F 04 04F8 E5 AD B0 9E 05B8 E9 60 85 A9
0060 80 52 47 55 02BC 85 A9 60 B1 037C AB 84 AF 60 043C 20 68 03 86 04FC 90 96 A5 A8 05BC E6 AE 4C A0
0200 A0 00 B1 AA 02C0 A2 85 A9 60 0380 A2 06 20 42 0440 AD B1 AA 29 0500 4A 90 DA 18 05C0 04 18 A5 A9
0204 A2 00 D5 00 02C4 A5 A9 91 A0 0384 03 A5 AB 91 0444 38 4A 4A AA 0504 90 D8 A5 A9 05C4 69 06 85 A9
0208 F0 25 E8 E0 02C8 60 A5 A9 91 0388 A6 20 42 03 0448 B5 45 85 F0 0508 25 AD 85 A9 05C8 A5 A9 29 0F
020C 1B 90 F7 B1 02CC A2 60 EA 20 038C A5 AA 91 A6 044C B5 46 85 F1 050C 20 12 03 4C 05CC C9 0A B0 F1
0210 AA 35 00 D5 02D0 D2 02 B1 A6 0390 4C 76 03 20 0450 6C F0 00 20 0510 A0 04 A5 A9 05D0 A5 A8 4A 90
0214 0F F0 18 E8 02D4 AA B9 A4 00 0394 68 03 B1 F0 0454 35 04 B1 F0 0514 45 AD 85 A9 05D4 06 A5 A9 20
0218 E0 2A 90 F3 02D8 91 A6 96 A4 0398 85 A4 C8 B1 0458 85 AD 4C 41 0518 20 12 03 4C 05D8 E3 05 EA A5
021C 20 F0 E9 A9 02DC C8 60 A5 A4 039C F0 85 A5 60 045C 04 20 35 04 051C 9A 04 A5 A9 05DC A9 C9 A0 B0
0220 45 20 7A E9 02E0 85 A6 A5 A5 03A0 20 68 03 B1 0460 B1 F0 48 20 0520 05 AD 4C 16 05E0 F4 90 DB 69
0224 A5 AB 20 46 02E4 85 A7 60 20 03A4 F0 85 A9 60 0464 18 04 68 91 0524 05 45 A9 48 05E4 5F 85 A9 4C
0228 EA A5 AA 4C 02E8 EA 02 B9 A4 03A8 20 68 03 20 0468 F0 60 20 68 0528 20 DD 04 68 05E8 FD 02 EA EA
022C 10 06 EA 8A 02EC 00 B6 A2 96 03AC AE 03 B9 A4 046C 03 8A 48 20 052C 85 A9 60 20 05EC EA 08 A9 00
0230 A2 00 D5 39 02F0 A4 99 A2 00 03B0 00 91 F0 C8 0470 18 04 68 91 0530 18 04 B1 F0 05F0 85 AE A5 A9
0234 90 03 E8 10 02F4 C8 60 A5 A9 03B4 60 20 68 03 0474 F0 60 18 4C 0534 48 29 0F C9 05F4 4C 7A 04 08
0238 F9 48 B5 3F 02F8 49 FF 85 A9 03B8 A5 A9 91 F0 0478 ED 05 48 65 0538 0F 20 01 06 05F8 A9 FF 85 AE
023C 85 AF 68 0A 02FC 60 A5 A8 09 03BC 60 20 2C 03 047C AD 85 A9 20 053C 68 18 69 01 05FC A5 A9 4C E1
0240 AA BD 5D 02 0300 03 85 A8 60 03C0 E0 06 90 02 0480 10 03 68 28 0540 91 F0 AA A5 0600 04 D0 06 A5
0244 85 F0 BD 5E 0304 A5 A8 49 01 03C4 E8 E8 60 20 0484 29 0F 48 A5 0544 A9 48 86 A9 0604 A8 09 12 D0
0248 02 85 F1 20 0308 85 A8 60 A5 03C8 BD 03 B5 A0 0488 AD 29 0F 85 0548 20 A0 04 68 0608 04 A5 A8 29
024C 5A 02 A5 AA 030C A9 0A 26 A9 03CC 48 B5 A1 48 048C AD 68 65 AD 054C 85 A9 60 20 060C C7 85 A8 60
0250 18 65 AF 85 0310 B0 EB A5 A8 03D0 A2 06 20 42 0490 29 10 F0 06 0550 18 04 B1 F0 0610 20 46 EA 4C
0254 AA 90 02 E6 0314 29 D6 85 A8 03D4 03 68 91 A6 0494 A5 A8 09 12 0554 38 E9 01 48 0614 62 03 E9 44
0258 AB 60 6C F0 0318 60 A5 A9 4A 03D8 20 42 03 68 0498 D0 04 A5 A8 0558 29 0F C9 0F
```

Bild 1. Hex-Dump des Simulatorprogramms. An den Adressen 0000...0060 sowie 025D...02B0 befinden sich Codetabellen. Die Simulatorroutine (Start 0200) ist als Unterprogramm ausgeführt

```
0F00 A9 LDA #00
0F02 85 STA AA
0F04 A9 LDA #08
0F06 85 STA AB
0F08 20 JSR 0200
0F0B 4C JMP 0F08
```

Bild 2. Einfaches 6502-Programm zum Ausführen von 8080-Maschinenprogrammen, die an der Adresse 0800 beginnen. Es ruft das Unterprogramm in Bild 1 zyklisch auf

also etwas über 1 KByte. Die verwendeten System-Monitoradressen nennt *Tabelle 1*. Die Ausführung der Befehle erfolgt genauso wie beim 8080, bis auf wenige Ausnahmen. So sind die Befehle RST 0...7, DI und EI nicht vorhanden, da sie der Interruptverarbeitung dienen, die bei einer softwaremäßigen Simulation nicht in Betracht kommt.

Andere Funktionen haben die Befehle OUT, IN und HLT erhalten. Bei OUT und IN wird das Argument als niederwertiges Adreßbyte interpretiert. Das höherwertige Byte dazu ist immer 06. Es erfolgt ein Sprung auf die angegebene Adresse, wo eine Ein- oder Ausgaberoutine in der 6502-Sprache steht. *Tabelle 2* gibt dafür ein Beispiel.

Bei HLT wird die Ausführung des Programms abgebrochen, und es erfolgt die Rückkehr zum System-Monitor.

Dasselbe geschieht, wenn der Simulator auf einen Code stößt, der nicht im Befehlssatz des 8080 enthalten ist, oder auf RST, DI und EI. In diesen Fällen wird zusätzlich die Adresse des fehlerhaften Bytes ausgegeben, d. h. der augenblickliche Stand des 8080-PC.

Der Simulator ist als Unterprogramm ausgeführt. Bei jedem Aufruf wird nur ein einziger Befehl ausgeführt. Daher ist es sinnvoll, einen kleinen Monitor zu schreiben, der nicht nur die Abarbeitung von Programmen, sondern auch das Setzen des Stackpointers und die Anzeige der 8080-Register übernimmt. Auf die Wiedergabe eines solchen Monitors wurde hier verzichtet, um nicht den universellen Charakter des Simulators für alle 6502-Systeme zu zerstören. In dieser Form läuft er nämlich ohne Änderungen auf allen Systemen, wenn sie nur über ausreichenden Speicherraum verfügen, da der System-Monitor nur bei der Fehlermeldung bemüht wird.

Soll das Programm in einen anderen Speicherbereich transferiert werden, so müssen auch die Tabellenwerte von 025D bis 02B0 und von 0045 bis 0054 entsprechend geändert werden; sie enthalten Sprungadressen.

Die Ausführungszeiten des Simulators liegen bei 117...1259 µs (Taktfrequenz: 1 MHz). Im Durchschnitt ist also etwa mit einer Verhundertfachung der 8080-Befehlszyklen zu rechnen. Sehr lange und zeitkritische 8080-Programme sollten daher lieber für den 6502 umgeschrieben werden, will man nicht eine extrem lange Laufzeit in Kauf nehmen. Zum Umschreiben aber ist das Verständnis des Programmablaufs erforderlich, und hierbei kann wiederum der Simulator eine wertvolle Hilfe sein.

```
0800 21 01 07  MVI HL,0701
0803 3A 00 07  LDA 0700
0806 86        ADD (HL)
0807 27        DAA
0808 32 02 07  STA 0702
0808 76        HLT
```

Bild 3. Dieses 8080-Programm addiert die Inhalte der Speicherzellen 0700 und 0701 und speichert das dezimale Ergebnis bei 0702 ab

Um den Einsatz der Simulator-Routine zu verdeutlichen, zeigt *Bild 2* ein kleines Programm, das den 8080-Programmzähler auf 0800 setzt und den dort stehenden 8080-Maschinencode abarbeitet, bis er auf HLT oder einen anderen Stopbefehl trifft. *Bild 3* enthält – wiederum als Beispiel – ein kurzes 8080-Programm, das den Inhalt der Adresse 0700 zum Inhalt der Adresse 0701 addiert, in den BCD-Code umformatiert (DAA) und bei 0702 abspeichert. Das Programm läuft in dieser Form unverändert auf dem AIM-65 und auf dem KIM-1, wobei allerdings jeweils die 4-KByte-Version vorhanden sein muß.

Literatur

[1] 8080-Operationscodes. FUNKSCHAU 1979, Heft 26.

[2] Der Individualisten-Prozessor. Sonderheft Hobbycomputer 2, Franzis-Verlag, München 1979.

Tabelle 1: Systemadressen

Adr.	*AIM-65*	*KIM-1*	
021C	20 F0 E9	20 2F 1E	CRLF
0221	20 7A E9	20 A0 1E	OUTCH
0226	20 46 EA	20 3B 1E	OUTBYT
0610	20 46 EA	20 3B 1E	OUTBYT
0365	4C 82 E1	4C 4F 1C	START

Tabelle 2: Routine für „OUT 16"

Adr.	*AIM-65*	*KIM-1*
0616	A5 A9	A5 A9
0618	4C 46 EA	4C 3B 1E

(Der Akku des 8080 wird auf dem AIM-Display bzw. dem KIM-Terminal in Form zweier Hex-Ziffern ausgegeben.)

Meine Meinung . . .

Daß man in der FUNKSCHAU ein Programm findet, mit dem die CPU 6502 den guten, alten 8080 simulieren kann, hat einen meiner Bekannten zu der Äußerung hingerissen: „Da kann man mal sehen, was der 6502 für ein mieser Prozessor sein muß, wenn seine Benutzer schon versuchen, eine andere CPU damit zu simulieren..." In der Tat gibt es eine Menge von 8080-Programmen, die auch für 6502-Besitzer interessant sind, und die CPU 8080 hat einige Befehle, von denen der 6502 nur träumen kann, z.B. zur Parity-Errechnung. Die Diskussion, welcher Mikroprozessor der beste ist (wenn es einen solchen überhaupt gibt), nimmt manchmal aber leider polemische Formen an. Seltsamerweise ist das Gesprächsthema Nr. 1 für zwei Mikrocomputer-Freunde immer noch, wer von beiden die bessere CPU besitzt! Dabei kennt meist keiner den Prozessor des anderen genau genug, um ein objektives Urteil fällen zu können – das ist bei der Komplexität der Befehlssätze ja auch kaum möglich. Vielleicht ist aber unser 8080-Simulationsprogramm ein kleiner Schritt zum besseren Verständnis – wenn auch nur in einer Richtung: Die 6502-Besitzer werden erkennen, daß man auch mit dem 8080-Befehlssatz etwas anfangen kann, wenn es auch einiger Umgewöhnung bedarf. Wer schreibt ein 8080-Programm zur Simulation des 6502?

Herwig Feichtinger

Michael Bleistein

Z-80-Assembler in Basic

2. Teil

Der einzulesende Datenfile wird durch ein unabhängiges Programm generiert. Das Listing dieses Hilfsprogramms zur Erzeugung einer Z-80-Codetabelle zeigt *Bild 4*. Wesentlich für die korrekte Arbeitsweise des Assemblers ist die genaue Einhaltung des Tabellenformats. Einzelheiten kann man *Tabelle 2* entnehmen. Weiter wird in diesem Programmteil der Name von Ein- und Ausgabefile abgefragt.

Die Zeilen 220 bis 870 ermöglichen es, einen Sourcefile zu erstellen und zu editieren. Falls ein Editor in der eigenen Systemsoftware bereits vorhanden ist, kann dieser Teil entfallen. Sollte die Speicherkapazität des Systems zu gering sein, so empfiehlt es sich, diesen Abschnitt als separates Programm zu schreiben und dann nur die jeweils benötigten Segmente einzulesen.

Die Eingabe eines Sourcefiles erfolgt formatfrei; es sind lediglich zwischen Label und Befehl sowie zwischen Befehl und Kommentar Strichpunkte zur Trennung einzusetzen. Wird eine Zeile ohne ein Label eingegeben, so muß der Befehl von einem Strichpunkt angeführt werden. Der Eingabemodus wird durch den Pseudobefehl „END" verlassen.

Zum Editieren sind drei Befehle vorhanden:

I =	Insert	Einfügen einer Zeile
D =	Delete	Löschen einer Zeile
L =	List	Formatiertes Listen des Sourcefile

Der Editor ist zeilenorientiert, d. h. es können nur ganze Zeilen eingeschoben oder gelöscht werden. Durch Eingabe von „E" (Exit) wird der Editor verlassen, und die Assemblierung beginnt. Da das Programm abfragt, ob eine Eingabe oder eine Editiergang erfolgen soll, ist es möglich, schon bestehende Sourcefiles unter Umgehung beider Routinen zu assemblieren.

Die Zeilen 890 bis 1330 besorgen den ersten Assemblierungsdurchgang. Hier wird der mnemonische Code in den Objectcode umgesetzt und eine Liste aller symbolischen Adressen erstellt. Können einzelne Zeilen nicht interpretiert werden, erfolgt eine Fehlermeldung, die auf dem File „FEHLIS.ASS" gespeichert wird. Eine Fehlermeldung führt nicht zum Abbruch der Assemblierung.

Die Zeilen 1350 bis 1690 stellen den zweiten Assemblierungsdurchgang dar. In diesem Abschnitt werden symbolische Operanden durch deren absolutes Äquivalent ersetzt.

Die Ausgabe erfolgt in den Zeilen 1710 bis 1900. Hier werden Source- und Objektfile auf einen Drucker oder ein Terminal ausgegeben. Es folgt die Fehlerliste und eine Aufstellung aller verwendeten symbolischen Adressen. Auch dieser Abschnitt kann als getrenntes Programm geschrieben und geladen werden, falls es erforderlich ist.

Die Zeilen 1920 bis 1970 sind ein Unterprogramm zum Separieren einer hexadezimalen Konstante aus dem Befehlscode, die Zeilen 1990 bis 2040 ein Unterprogramm zum Separieren eines Labels aus dem Befehlscode, die Zeilen 2060 bis 2130 ein Unterprogramm zur Umrechnung von Hexadezimal- in Dezimalzahlen. Die Hexadezimalzahl wird als Variable H$ übergeben, die Ausgabe erfolgt über D0. In den Zeilen 2150 bis 2240 steht ein Unterprogramm zur Umwandlung von Dezimal- in Hexadezimalzahlen. Die Übergabe der Variablen erfolgt sinngemäß wie oben.

Das Unterprogramm in den Zeilen 2260 bis 2530 übersetzt mnemonische Abkürzungen in den maschinenlesbaren Objectcode. Die Länge eines Maschinenbefehls ist dabei durch seine Lage in der Codetabelle festgelegt. Innerhalb dieser Grenzen empfiehlt es sich, die Befehle nach der Häufigkeit ihres Auftretens anzuordnen. Durch geschickte Plazierung der Befehle kann Programmlaufzeit eingespart werden, da die Codetabelle immer von Anfang an abgesucht wird.

Um die Unterscheidung von symbolischen Adressen und Hexadezimalkonstanten zu vereinfachen, müssen beide in spezielle Begrenzungszeichen gesetzt werden. Wird das Label „LABEL" als Operand verwendet, so muß es als $LABEL$, die Hexadezimalkonstante „F80E" als #F80E# geschrieben werden. Diese Festsetzung erschwert zwar die Eingabe, hat aber zur Folge, daß die Programmlaufzeit verkürzt wird und vollkommene Freiheit in der Zeichenauswahl für die symbolischen Adressen besteht. Die maximale Länge eines Labels beträgt sechs Buchstaben, wobei auch Zahlen und Sonderzeichen mit Ausnahme von $ und # zugelassen sind.

Ein Beispiel

Bild 5 zeigt den Ausdruck eines Assemblierungsdurchgangs. Es wurde ein Programm zur Ein- und Ausgabe eines ASCII-Zeichens über einen seriellen Port vom Typ 8251 (Portadresse 04/06) assembliert. Im ersten Abschnitt wird nach der Eingabe von Ein- und Ausgabefile der Sourcecode eingegeben. Im Edit-Mode wird die Eingabe durch das „L"-Kommando überprüft. Mit Hilfe der Befehle „I" und „D" sind jetzt noch Korrekturen möglich. Danach startet man durch „E" die Assemblierung. Nach ca. 30 s erfolgt die Ausgabe.

Betrieb ohne Plattenspeicher

Da sicher viele Anwender des Programms nicht in der Lage sein werden, mit einer Platteneinheit als Speichermedium zu arbeiten, sollen noch einige Anregungen zum Betrieb mit Kassettenrecorder und Halbleiterspeicher gegeben werden.

Tabelle 2: Format des Code-File

BEFEHL	HEXCODE	CR	LF

Beispiele:
DEC H/25 <CR> <LF>
LD (IX+)/DD35 <CR> <LF>
BIT 5 (IX+)/DDCBHH6E <CR> <LF>

Tabelle 3: Fehlermeldungen

ILL CODE AT LINE XXXX
Fehlerhafter Befehlskode in Zeile XXXX

LBL NFD AT LINE XXXX
Label das in Zeile XXXX auftritt ist nicht definiert

JR OOR AT LINE XXXX
Sprungbereich eines relativen Sprungs überschritten

```
10 REM ************************************************************
20 REM ***  HILFSPROGRAMM ZUR ERZEUGUNG EINES DATENFILES  **********
30 REM ***                MIT Z-80 MNEMONICS              **********
40 REM ************************************************************
50 REM
60 REM MICHAEL BLEISTEIN ; 28.03.1979
70 REM
80 OPEN "DATA.ASS" FOR OUTPUT AS FILE #1
90 FOR I=1 TO 694
100 READ O$
110 PRINT #1,O$
120 NEXT I
130 DATA "ADC A,(HL)/8E","ADC A,A/8E","ADC A,B/88"
140 DATA "ADC A,C/89","ADC A,D/8A","ADC A,E/8B","ADC A,F/8C"
150 DATA "ADC A,L/8D"
160 DATA "ADD A,(HL)/86","ADD A,A/87","ADD A,B/80"
170 DATA "ADD A,C/81","ADD A,D/82","ADD A,E/83","ADD A,H/84"
180 DATA "ADD A,L/85","ADD HL,BC/09","ADD HL,DE/19"
190 DATA "ADD HL,HL/29","ADD HL,SP/39"
200 DATA "AND (HL)/A6","AND A/A7","AND B/A0","AND C/A1"
210 DATA "AND D/A2","AND E/A3","AND H/A4","AND L/A5"
220 DATA "CCF/3F","CP (HL)/BE","CP A/BF","CP B/B8"
230 DATA "CP C/B9","CP D/BA","CP E/BB","CP H/BC","CP L/BD"
240 DATA "CPL/2F","DAA/27","INC HL/23"
250 DATA "DEC (HL)/35","DEC A/3D","DEC B/05","DEC BC/0B"
260 DATA "DEC C/0D","DEC DE/1B","DEC D/15","DEC E/1D"
270 DATA "DEC H/25","DEC HL/2B","DEC L/2D","DEC SP/3B"
280 DATA "DI/F3","EI/FB","EX (SP),HL/E3","EX AF,AF'/08"
290 DATA "EX DE,HL/EB","EXX/D9","HALT/76","INC (HL)/34"
300 DATA "INC A/3C","INC B/04","INC C/0C","INC BC/03"
310 DATA "INC D/14","INC DE/13","INC E/1C","INC H/24"
320 DATA "INC L/2C","INC SP/33","JP (HL)/E9"
330 DATA "LD (BC),A/02","LD (DE),A/12","LD (HL),A/77"
340 DATA "LD (HL),B/70","LD (HL),C/71","LD (HL),D/72"
350 DATA "LD (HL),E/73","LD (HL),H/74","LD (HL),L/75"
360 DATA "LD A,(BC)/0A","LD A,(DE)/1A","LD A,(HL)/7E"
370 DATA "LD A,A/7F","LD A,B/78","LD A,C/79","LD A,D/7A"
380 DATA "LD A,E/7B","LD A,H/7C","LD A,L/7D","LD B,(HL)/46"
390 DATA "LD B,A/47","LD B,B/40","LD B,C/41","LD B,D/42"
400 DATA "LD B,E/43","LD B,H/44","LD B,L/45","LD C,(HL)/4E"
410 DATA "LD C,A/4F","LD C,B/48","LD C,C/49","LD C,D/4A"
420 DATA "LD C,E/4B","LD C,H/4C","LD C,L/4D","LD D,(HL)/56"
430 DATA "LD D,A/57","LD D,B/50","LD D,C/51","LD D,D/52"
440 DATA "LD D,E/53","LD D,H/54","LD D,L/55","LD E,(HL)/5E"
450 DATA "LD E,A/5F","LD E,B/58","LD E,C/59","LD E,D/5A"
460 DATA "LD E,E/5B","LD E,H/5C","LD E,L/5D","LD H,(HL)/66"
470 DATA "LD H,A/67","LD H,B/60","LD H,C/61","LD H,D/62"
480 DATA "LD H,E/63","LD H,H/64","LD H,L/65","LD L,(HL)/6E"
490 DATA "LD L,A/6F","LD L,B/68","LD L,C/69","LD L,D/6A"
500 DATA "LD L,E/6B","LD L,H/6C","LD L,L/6D","LD SP,HL/F9"
510 DATA "NOP/00","OR (HL)/B6","OR A/B7","OR B/B0"
520 DATA "OR C/B1","OR D/B2","OR E/B3","OR H/B4","OR L/B5"
530 DATA "POP AF/F1","POP BC/C1","POP DE/D1"
540 DATA "POP HL/E1","PUSH AF/F5","PUSH BC/C5","PUSH DE/D5"
550 DATA "PUSH HL/E5","RET/C9","RET C/D8","RET M/F8"
560 DATA "RET NC/D0","RET NZ/C0","RET P/F0","RET PE/E8"
570 DATA "RET PO/E0","RET Z/C8"
580 DATA "RLA/17","RLCA/07","RRA/1F","RRCA/0F","RST 0/C7"
590 DATA "RST 10/D7","RST 18/DF","RST 20/E7","RST 28/EF"
600 DATA "RST 30/F7","RST 38/FF","RST 8/CF","SBC A,(HL)/9E"
610 DATA "SBC A,A/9F","SBC A,B/98","SBC A,C/99","SBC A,D/9A"
620 DATA "SBC A,E/9B","SBC A,H/9C","SBC A,L/9D","SCF/37"
630 DATA "SUB (HL)/96","SUB A/97","SUB B/90","SUB C/91"
640 DATA "SUB D/92","SUB E/93","SUB H/94","SUB L/95"
650 DATA "XOR (HL)/AE","XOR A/AF","XOR B/A8","XOR C/A9"
660 DATA "XOR D/AA","XOR E/AB","XOR H/AC","XOR L/AD"
670 DATA "ADC A,/CE"
680 DATA "ADC HL,BC/ED4A","ADC HL,DE/ED5A","ADC HL,HL/ED6A"
690 DATA "ADC HL,SP/ED7A","ADD A,/C6","ADD IX,BC/DD09"
700 DATA "ADD IX,DE/DD19","ADD IX,IX/DD29","ADD IX,SP/DD39"
710 DATA "ADD IY,BC/FD09","ADD IY,DE/FD19","ADD IY,IY/FD29"
720 DATA "ADD IY,SP/FD39"
730 DATA "AND/E6","CP/FE","CPD/EDA9","CPDR/EDB9","CPI/EDA1"
740 DATA "CPIR/EDB1"
750 DATA "DEC IX/DD2B","DEC IY/FD2B","DJNZ/10"
760 DATA "EX (SP),IX/DD3E","EX (SP),IY/FDE3","IM 0/ED46"
770 DATA "IM 1/ED56","IM 2/ED5E","IN A,(C)/ED78","IN A,()/DB"
780 DATA "IN B,(C)/ED40","IN C,(C)/ED48","IN D,(C)/ED50"
790 DATA "IN E,(C)/ED58","IN H,(C)/ED60","IN L,(C)/ED68"
800 DATA "BIT 0,(HL)/CB46","BIT 0,A/CB47","BIT 0,B/CB40"
810 DATA "BIT 0,C/CB41","BIT 0,D/CB42","BIT 0,E/CB43"
820 DATA "BIT 0,H/CB44","BIT 0,L/CB45","BIT 1,(HL)/CB4E"
830 DATA "BIT 1,A/CB4F","BIT 1,B/CB48","BIT 1,C/CB49"
840 DATA "BIT 1,D/CB4A","BIT 1,E/CB4B","BIT 1,H/CB4C"
850 DATA "BIT 1,L/CB4D","BIT 2,(HL)/CB56","BIT 2,A/CB57"
860 DATA "BIT 2,B/CB50","BIT 2,C/CB51","BIT 2,D/CB52"
870 DATA "BIT 2,E/CB53","BIT 2,H/CB54","BIT 2,L/CB55"
880 DATA "BIT 3,(HL)/CB5E","BIT 3,A/CB5F","BIT 3,B/CB58"
890 DATA "BIT 3,C/CB59","BIT 3,D/CB5A","BIT 3,E/CB5B"
900 DATA "BIT 3,H/CB5C","BIT 3,L/CB5D","BIT 4,(HL)/CB66"
910 DATA "BIT 4,A/CB67","BIT 4,B/CB60","BIT 4,C/CB61"
920 DATA "BIT 4,D/CB62","BIT 4,E/CB63","BIT 4,H/CB64"
930 DATA "BIT 4,L/CB65","BIT 5,(HL)/CB6E","BIT 5,A/CB6F"
940 DATA "BIT 5,B/CB68","BIT 5,C/CB69","BIT 5,D/CB6A"
950 DATA "BIT 5,E/CB6B","BIT 5,H/CB6C","BIT 5,L/CB6D"
960 DATA "BIT 6,(HL)/CB76","BIT 6,A/CB77","BIT 6,B/CB70"
970 DATA "BIT 6,C/CB71","BIT 6,D/CB72","BIT 6,E/CB73"
980 DATA "BIT 6,H/CB74","BIT 6,L/CB75","BIT 7,(HL)/CB7E"
990 DATA "BIT 7,A/CB7F","BIT 7,B/CB78","BIT 7,C/CB79"
1000 DATA "BIT 7,D/CB7A","BIT 7,E/CB7B","BIT 7,H/CB7C"
1010 DATA "BIT 7,L/CB7D"
1020 DATA "INC IX/DD23","INC IY/FD23","IND/EDAA","INDR/EDBA"
1030 DATA "INI/EDA2","INIR/EDB2","JP (IX)/DDE9","JP (IY)/FDE9"
1040 DATA "JR C,/38","JR/18","JR NC,/30","JR NZ,/20"
1050 DATA "JR Z,/28"
1060 DATA "LD (HL),/36","LD A,I/ED57","LD A,/3E","LD B,/06"
1070 DATA "LD C,/0E","LD D,/16","LD E,/1E","LD H,/26"
1080 DATA "LD I,A/ED47","LD L,/2E","LD SP,IX/DDF9"
1090 DATA "LD SP,IY/FDF9","LDD/EDA8","LDDR/EDB8","LDI/EDA0"
1100 DATA "LDIR/EDB0","NEG/ED44","OR/F6","OTDR/EDBB"
1110 DATA "OTIR/EDB3","OUT (C),A/ED79","OUT (C),B/ED41"
1120 DATA "OUT (C),C/ED49","OUT (C),D/ED51","OUT (C),E/ED59"
1130 DATA "OUT (C),H/ED61","OUT (C),L/ED69","OUT (),A/D3"
1140 DATA "OUTD/EDAB","PUSH IX/DDE5","OUTI/EDA3","LD A,R/ED5F"
1150 DATA "POP IX/DDE1","EX (SP),IX/DDE3","POP IY/FDE1","PUSH IX/DD5E"
1160 DATA "PUSH IY/FDE5","RETI/ED4D","RETN/ED45"
1170 DATA "RES 0,(HL)/CB86","RES 0,A/CB87","RES 0,B/CB80"
1180 DATA "RES 0,C/CB81","RES 0,D/CB82","RES 0,E/CB83"
1190 DATA "RES 0,L/CB85","RES 0,H/CB84","RES 1,(HL)/CB8E"
1200 DATA "RES 1,A/CB8F","RES 1,B/CB88","RES 1,C/CB89"
1210 DATA "RES 1,D/CB8A","RES 1,E/CB8B","RES 1,H/CB8C"
1220 DATA "RES 1,L/CB8D","RES 2,(HL)/CB96","RES 2,A/CB97"
1230 DATA "RES 2,B/CB90","RES 2,C/CB91","RES 2,D/CB92"
1240 DATA "RES 2,E/CB93","RES 2,H/CB94","RES 2,L/CB95"
1250 DATA "RES 3,(HL)/CB9E","RES 3,A/CB9F","RES 3,B/CB98"
1260 DATA "RES 3,C/CB99","RES 3,D/CB9A","RES 3,E/CB9B"
```

```
1270 DATA "RES 3,H/CB9C","RES 3,L/CB9D","RES 4,(HL)/CBA6"
1280 DATA "RES 4,A/CBA7","RES 4,B/CBA0","RES 4,C/CBA1"
1290 DATA "RES 4,D/CBA2","RES 4,E/CBA3","RES 4,H/CBA4"
1300 DATA "RES 4,L/CBA5"
1310 DATA "RES 5,(HL)/CBAE","RES 5,A/CBAF","RES 5,B/CBA8"
1320 DATA "RES 5,C/CBA9","RES 5,D/CBAA","RES 5,E/CBAB"
1330 DATA "RES 5,H/CBAC","RES 5,L/CBAD","RES 6,(HL)/CBB6"
1340 DATA "RES 6,A/CBB7","RES 6,B/CBB0","RES 6,C/CBB1"
1350 DATA "RES 6,D/CBB2","RES 6,E/CBB3","RES 6,H/CBB4"
1360 DATA "RES 6,L/CBB5","RES 7,(HL)/CBBE","RES 7,A/CBBF"
1370 DATA "RES 7,B/CBB8","RES 7,C/CBB9","RES 7,D/CBBA"
1380 DATA "RES 7,E/CBBB","RES 7,H/CBBC","RES 7,L/CBBD"
1390 DATA "RL (HL)/CB16","RL A/CB17","RL B/CB10","RL C/CB11"
1400 DATA "RL D/CB12","RL E/CB13","RL H/CB14","RL L/CB15"
1410 DATA "RLC (HL)/CB06","RLC A/CB07","RLC B/CB00"
1420 DATA "RLC C/CB01","RLC D/CB02","RLC E/CB03","RLC H/CB04"
1430 DATA "RLC L/CB05","RLD/ED6F","RR (HL)/CB1E","RR A/CB1F"
1440 DATA "RR B/CB18","RR C/CB19","RR D/CB1A","RR E/CB1B"
1450 DATA "RR H/CB1C","RR L/CB1D","RRC (HL)/CB0E","RRC A/CB0F"
1460 DATA "RRC B/CB08","RRC C/CB09","RRC D/CB0A","RRC E/CB0B"
1470 DATA "RRC H/CB0C","RRC L/CB0D","RRD/ED67","SBC A,/DE"
1480 DATA "SBC HL,BC/ED42","SBC HL,DE/ED52","SBC HL,HL/ED62"
1490 DATA "SBC HL,SP/ED72"
1500 DATA "SET 0,(HL)/CBC6","SET 0,A/CBC7","SET 0,B/CBC0"
1510 DATA "SET 0,C/CBC1","SET 0,D/CBC2","SET 0,E/CBC3"
1520 DATA "SET 0,H/CBC4","SET 0,L/CBC5","SET 1,(HL)/CBCE"
1530 DATA "SET 1,A/CBCF","SET 1,B/CBC8","SET 1,C/CBC9"
1540 DATA "SET 1,D/CBCA","SET 1,E/CBCB","SET 1,H/CBCC"
1550 DATA "SET 1,L/CBCD","SET 2,(HL)/CBD6","SET 2,A/CBD7"
1560 DATA "SET 2,B/CBD0","SET 2,C/CBD1","SET 2,D/CBD2"
1570 DATA "SET 2,E/CBD3","SET 2,H/CBD4","SET 2,L/CBD5"
1580 DATA "SET 3,(HL)/CBDE","SET 3,A/CBDF","SET 3,B/CBD8"
1590 DATA "SET 3,C/CBD9","SET 3,D/CBDA","SET 3,E/CBDB"
1600 DATA "SET 3,H/CBDC","SET 3,L/CBDD","SET 4,(HL)/CBE6"
1610 DATA "SET 4,A/CBE7","SET 4,B/CBE0","SET 4,C/CBE1"
1620 DATA "SET 4,D/CBE2","SET 4,E/CBE3","SET 4,H/CBE4"
1630 DATA "SET 4,L/CBE5"
1640 DATA "SET 5,(HL)/CBEE","SET 5,A/CBEF","SET 5,B/CBE8"
1650 DATA "SET 5,C/CBE9","SET 5,D/CBEA","SET 5,E/CBEB"
1660 DATA "SET 5,H/CBEC","SET 5,L/CBED","SET 6,(HL)/CBF6"
1670 DATA "SET 6,A/CBF7","SET 6,B/CBF0","SET 6,C/CBF1"
1680 DATA "SET 6,D/CBF2","SET 6,E/CBF3","SET 6,H/CBF4"
1690 DATA "SET 6,L/CBF5","SET 7,(HL)/CBFE","SET 7,A/CBFF"
1700 DATA "SET 7,B/CBF8","SET 7,C/CBF9","SET 7,D/CBFA"
1710 DATA "SET 7,E/CBFB","SET 7,H/CBFC","SET 7,L/CBFD"
1720 DATA "SLA (HL)/CB26","SLA A/CB27","SLA B/CB20"
1730 DATA "SLA C/CB21","SLA D/CB22","SLA E/CB23","SLA H/CB24"
1740 DATA "SLA L/CB25","SRA (HL)/CB2E","SRA A/CB2F"
1750 DATA "SRA B/CB28","SRA C/CB29","SRA D/CB2A","SRA E/CB2B"
1760 DATA "SRA H/CB2C","SRA L/CB2D","SRL (HL)/CB3E"
1770 DATA "SRL A/CB3F","SRL B/CB38","SRL C/CB39","SRL D/CB3A"
1780 DATA "SRL E/CB3B","SRL H/CB3C","SRL L/CB3D","SUB/D6"
1790 DATA "XOR/EE"
1800 DATA "ADC A,(IX+)/DD8E","ADC A,(IY+)/FD8E"
1810 DATA "ADD A,(IX+)/DD86","ADD A,(IY+)/FD86"
1820 DATA "AND (IX+)/DDA6","AND (IY+)/FDA6","CALL C,/DC"
1830 DATA "CALL M,/FC","CALL NC,/D4","CALL NZ,/C4"
1840 DATA "CALL P,/F4","CALL PE,/EC","CALL PO,/E4"
1850 DATA "CALL Z,/CC","CALL/CD","CP (IX+)/DDBE","CP (IY+)/FDBE"
1860 DATA "DEC (IX+)/DD35","DEC (IY+)/FD35","INC (IX+)/DD34"
1870 DATA "INC (IY+)/FD34"
1880 DATA "JP C,/DA","JP M,/FA","JP NC,/D2","JP/C3"
1890 DATA "JP NZ,/C2","JP P,/F2","JP PE,/EA","JP PO,/E2"
1900 DATA "JP Z,/CA"
1910 DATA "LD (IX+),A/DD77","LD (IX+),B/DD70","LD (IX+),C/DD71"
1920 DATA "LD (IX+),D/DD72","LD (IX+),E/DD73","LD (IX+),H/DD74"
1930 DATA "LD (IX+),L/DD75","LD (IY+),A/FD77","LD (IY+),B/FD70"
1940 DATA "LD (IY+),C/FD71","LD (IT+),D/FD72","LD (IY+),E/FD73"
1950 DATA "LD (IY+),H/FD74","LD (IY+),L/FD75","LD (),A/32"
1960 DATA "LD (),HL/22","LD A,(IX+)/DD7E","LD A,(IY+)/FD7E"
1970 DATA "LD A,()/3A","LD B,(IX+)/DD46","LD B,(IT+)/FD46"
1980 DATA "LD BC,/01","LD C,(IX+)/DD4E","LD C,(IY+)/FD4E"
1990 DATA "LD D,(IX+)/DD56","LD D,(IY+)/FD56","LD DE,/11"
2000 DATA "LD E,(IX+)/DD5E","LD E,(IY+)/FD5E","LD H,(IX+)/DD66"
2010 DATA "LD H,(IY+)/FD66","LD HL,()/2A","LD HL,/21"
2020 DATA "LD L,(IX+)/DD6E","LD L,(IY+)/FD6E","LD SP,/31"
2030 DATA "OR (IX+)/DDB6","OR (IY+)/FDB6"
2040 DATA "SBC A,(IX+)/DD9E","SBC A,(IY+)/FD9E"
2050 DATA "SUB (IX+)/DD96","SUB (IY+)/FD96","XOR (IX+)/DDAE"
2060 DATA "XOR (IY+)/FDAE"
2070 DATA "BIT 0,(IX+)/DDCBHH46","BIT 0,(IY+)/FDCBHH46"
2080 DATA "BIT 1,(IX+)/DDCBHH4E","BIT 1,(IY+)/FDCBHH4E"
2090 DATA "BIT 2,(IX+)/DDCBHH56","BIT 2,(IY+)/FDCBHH56"
2100 DATA "BIT 3,(IX+)/DDCBHH5E","BIT 3,(IY+)/FDCBHH5E"
2110 DATA "BIT 4,(IX+)/DDCBHH66","BIT 4,(IY+)/FDCBHH66"
2120 DATA "BIT 5,(IX+)/DDCBHH6E","BIT 5,(IY+)/FDCBHH6E"
2130 DATA "BIT 6,(IX+)/DDCBHH76","BIT 6,(IY+)/FDCBHH76"
2140 DATA "BIT 7,(IY+)/FDCBHH7E","BIT 7,(IX+)/DDCBHH7E"
2150 DATA "LD (IX+)/DD36","LD (IY+)/FD36","LD (),BC/ED43"
2160 DATA "LD (),DE/ED53","LD (),IX/DD22","LD (),IY/FD22"
2170 DATA "LD (),SP/ED73","LD BC,()/ED4B","LD DE,()/ED5B"
2180 DATA "LD IX,()/DD2A","LD IX,/DD21","LD IY,()/FD2A"
2190 DATA "LD IY,/FD21","LD SP,()/ED7B","RES 0,(IX+)/DDCBHH86"
2200 DATA "RES 0,(IY+)/FDCBHH86","RES 1,(IX+)/DDCBHH8E"
2210 DATA "RES 1,(IY+)/FDCBHH8E","RES 2,(IX+)/DDCBHH96"
2220 DATA "RES 2,(IY+)/FDCBHH96","RES 3,(IX+)/DDCBHH9E"
2230 DATA "RES 3,(IY+)/FDCBHH9E","RES 4,(IX+)/DDCBHHA6"
2240 DATA "RES 4,(IY+)/FDCBHHA6","RES 5,(IX+)/DDCBHHAE"
2250 DATA "RES 5,(IY+)/FDCBHHAE","RES 6,(IX+)/DDCBHHB6"
2260 DATA "RES 6,(IY+)/FDCBHHB6","RES 7,(IX+)/DDCBHHBE"
2270 DATA "RES 7,(IY+)/FDCBHHBE","RL (IX+)/DDCBHH16"
2280 DATA "RL (IY+)/FDCBHH16","RLC (IX+)/DDCBHH06"
2290 DATA "RLC (IY+)/FDCBHH06","RR (IX+)/DDCBHH1E"
2300 DATA "RR (IY+)/FDCBHH1E","RRC (IX+)/DDCBHH0E"
2310 DATA "RRC (IY+)/FDCBHH0E","SET 0,(IX+)/DDCBHHC6"
2320 DATA "SET 0,(IY+)/FDCBHHC6","SET 1,(IX+)/DDCBHHCE"
2330 DATA "SET 1,(IY+)/FDCBHHCE","SET 2,(IX+)/DDCBHHD6"
2340 DATA "SET 2,(IY+)/FDCBHHD6","SET 3,(IX+)/DDCBHHDE"
2350 DATA "SET 3,(IY+)/FDCBHHDE","SET 4,(IX+)/DDCBHHE6"
2360 DATA "SET 4,(IY+)/FDCBHHE6","SET 5,(IX+)/DDCBHHEE"
2370 DATA "SET 5,(IY+)/FDCBHHEE","SET 6,(IX+)/DDCBHHF6"
2380 DATA "SET 6,(IY+)/FDCBHHF6","SET 7,(IX+)/DDCBHHFE"
2390 DATA "SET 7,(IY+)/FDCBHHFE","SLA (IX+)/DDCBHH26"
2400 DATA "SLA (IY+)/FDCBHH26","SRA (IX+)/DDCBHH2E"
2410 DATA "SRA (IY+)/FDCBHH2E","SRL (IX+)/DDCBHH3E"
2420 DATA "SRL (IY+)/FDCBHH3E"
```

Bild 4. Hilfsprogramm zur Erzeugung des Z-80-Codefiles

```
ADR  OBJ CODE  STMT LABEL   SOURCE CODE      COMMENT

0000           0001         ORG 4100         STARTADRESSE
4100 DB06      0002 INCHR   IN A,(#06#)      ABFRAGE STATUS
4102 CB57      0003         BIT 2,A
4104 28FA      0004         JR Z,$INCHR$     WARTEN READY
4106 3E17      0005         LD A,#17#        INITIALISIEREN
4108 D306      0006         OUT (#06#),A
410A DB06      0007 LOOP1   IN A,(#06#)      WARTEN INPUT
410C CB4F      0008         BIT 1,A
410E 28FA      0009         JR Z,$LOOP1$
4110 DB04      0010         IN A,(#04#)      ASCII INPUT
4112 C9        0011         RET
4113 322641    0012 OUTCHR  LD ($MEM$),A
4116 DB06      0013 LOOP2   IN A,(#06#)      STATUS ABFRAGEN
4118 CB57      0014         BIT 2,A
411A 28FA      0015         JR Z,$LOOP2$     INPUT ?
411C 3E31      0016         LD A,#31#        INITIALISIEREN
411E D306      0017         OUT (#06#),A
4120 3A2641    0018         LD A,($MEM$)     AKKU REFRESH
4123 D304      0019         OUT (#04#),A     ASCII OUTPUT
4125 C9        0020         RET
4126           0021 MEM     DB               ZWISCHENSPEICHER
4127           0022         END

FEHLER

LABEL LISTING

INCHR     4100
LOOP1     410A
OUTCHR    4113
LOOP2     4116
MEM       4126
```

Wird auf jedes externe Speichermedium verzichtet, so müssen eindimensionale Stringfelder die entsprechenden Files ersetzen. Die Codetabelle muß dann über READ-Anweisungen eingelesen werden. Nachteilig bei diesem Verfahren ist allerdings der hohe Speicherbedarf und damit der vergleichsweise geringe Umfang der noch zu verarbeitenden Sourcefiles. Der Vorteil ist die hohe Verarbeitungsgeschwindigkeit. Zu dieser Lösung wird man daher nur zu Demonstrationszwecken greifen.

Die zweite Möglichkeit ist der Betrieb des Programms mit einem oder zwei Kassettenrecordern. Je nach Anzahl der ansteuerbaren Kassettenrecorder wird man die speicherintensivsten Files auf einer Kassette anlegen. In erster Linie ist dies der Sourcefile. Alle anderen Files können wie oben angegeben als eindimensionale Stringfelder behandelt werden. Da der Assembler aber zwei Durchgänge benötigt, muß nach dem ersten zurückgespult werden.

Verwendung mit anderen CPU-Typen

Durch die verwendete Programmsprache kann der Assembler auf jedem System mit Standard-Basic-Interpreter betrieben werden. Es muß nur die jeweilige Codetabelle geladen werden. Zusätzlich müssen die Grenzen zwischen Gebieten unterschiedlicher Befehlslänge neu eingegeben werden (Zeile 2460). In Zeile 1500 bis 1520 werden relative Sprünge erkannt und entsprechend ausgewertet. Diese Zeilen müssen auch auf den jeweiligen CPU-Typ abgestimmt werden.

Literatur

Schmitt, G.: Einfacher Cross-Assembler für Mikroprozessoren; ELEKTRONIK 5/78, S. 69.

```
RSSZ80  18-JUN-79  BASIC V01-05

INPUTFILE?INOUT
OUTPUTFILE?INOUTO
INPUT MODE (Y/N)?Y

?;ORG 4100;STARTADRESSE
?INCHR;IN A,(#06#);ABFRAGE STATUS
?;BIT 2,A
?;JR Z,$INCHR$;WARTEN READY
?;LD A,#17#;INITIALISIEREN
?;OUT (#06#),A
?LOOP1;IN A,(#06#);WARTEN INPUT
?;BIT 1,A
?;JR Z,$LOOP1$
?;IN A,(#04#);ASCII INPUT
?;RET
?OUTCHR;LD ($MEM$),A
?LOOP2;IN A,(#06#);STATUS ABFRAGEN
?;BIT 2,A
?;JR Z,$LOOP2$;INPUT ?
?;LD A,#31#;INITIALISIEREN
?;OUT (#06#),A
?;LD A,($MEM$);AKKU REFRESH
?;OUT (#04#),A;ASCII OUTPUT
?;RET
?MEM;DB;ZWISCHENSPEICHEN
?;END

EDIT MODE (Y/N)?Y

I=INSERT;D=DELETE;E=EXIT;L=LIST
```

Bild 5. Das hier assemblierte Programm dient zur ASCII-Ein- und Ausgabe über einen seriellen Port vom Typ 8251

```
?L

INOUT.ASS        Z80 ASSEMBLER      V 1.0

0001           ORG 4100          STARTADRESSE
0002 INCHR     IN A,(#06#)       ABFRAGE STATUS
0003           BIT 2,A
0004           JR Z,$INCHR$      WARTEN READY
0005           LD A,#17#         INITIALISIEREN
0006           OUT (#06#),A
0007 LOOP1     IN A,(#06#)       WARTEN INPUT
0008           BIT 1,A
0009           JR Z,$LOOP1$
0010           IN A,(#04#)       ASCII INPUT
0011           RET
0012 OUTCHR    LD ($MEM$),A
0013 LOOP2     IN A,(#06#)       STATUS ABFRAGEN
0014           BIT 2,A
0015           JR Z,$LOOP2$      INPUT ?
0016           LD A,#31#         INITIALISIEREN
0017           OUT (#06#),A
0018           LD A,($MEM$)      AKKU REFRESH
0019           OUT (#04#),A      ASCII OUTPUT
0020           RET
0021 MEM       DB                ZWISCHENSPEICHER
0022           END
```

Computer zum Selbstbau

Das H-8-System von Heathkit

Eines der komfortabelsten Computersysteme zum Selbstbau ist derzeit wohl der H-8 von Heathkit nebst Peripherie: das Floppy-Disk-Laufwerk H-17, das Videoterminal H-9 und der Drucker H-14. Dieser Drucker läßt sich dank seiner RS-232-Schnittstelle übrigens auch mit anderen Computern problemlos verbinden und ist in erster Linie wegen seines geringen Preises interessant.

Bild 1. Das Videoterminal H-9

Doch zunächst zum Computer selbst. Der H-8 arbeitet mit der CPU 8080 und verfügt über zahlreiche Steckplätze für Speichererweiterungen, eine Floppy-Disk-Controller-Karte und Eingabe/Ausgabe-Erweiterungen. Ein Netzteil ist wie auch bei allen Peripheriegeräten eingebaut. Im Grundzustand lassen sich Programme auf einem externen Kassettenrecorder speichern; leider ist das in ROMs gespeicherte Monitorprogramm nicht allzu komfortabel, so daß es sich empfiehlt, entweder von der Kassette oder von der Floppy-Disk ein leistungsfähigeres Betriebssystem zu laden, z. B. das Debugger-Programm BUG-8. Es umfaßt 3 KByte und erlaubt das Einschreiben von Programmen, deren Prüfen im Einzelschritt-Betrieb und das Setzen von sog. Breakpoints. Der Texteditor TED-8 erlaubt bereits eine einfache Textverarbeitung zusammen mit dem Terminal H-9 *(Bild 1)*, leidet allerdings an dessen Einschränkung, keine Kleinbuchstaben verarbeiten zu können. Schließlich gibt es noch den 8080-Assembler HASL-8 und das Benton Harbor BASIC, das auch so nützliche Befehle wie BUILD, SCRATCH, PAUSE, PORT und MAX einschließt.

Für den Umgang mit dem Floppy-Disk-Speicher H-17 benötigt man das Disk-Operating-System HDOS, das normalerweise auf einer Floppy geliefert wird und über ein einfaches Bootstrap-Programm im ROM des H-8 in das System-RAM übernommen wird. Außer einem Diagnostik-Programm bietet es auch die Möglichkeit, bestimmte Files auf einer Diskette per Software vor dem Überschreiben zu schützen.

Der Drucker H-14

Der Heath-Drucker *(Bild 2)* ist auch für die Benutzer anderer Mikrocomputer-Systeme interessant – vor allem wegen seines geringen Preises als Bausatz – und wird hier etwas näher besprochen. Er benötigt vorgefaltetes, an den beiden Seiten gelochtes Papier mit 3...9,5 Zoll Breite. Die Verbindung zum Computer geschieht mit einer seriellen RS-232- oder 20-mA-Schnittstelle und einem Handshake-Busy-Signal; bei 110 Bd kann man auf die Handshake-Leitung verzichten. Die Übertragungsgeschwindigkeit kann 110...4800 Bd betragen, wobei ein 256-Zeichen-Bufferspeicher für die eingehenden ASCII-Zeichen zur Verfügung steht. Der Anschluß an beliebige Mikrocomputer-Systeme ist im Handbuch des Druckers ausführlich beschrieben.

Bestimmte Steuerzeichen werden im Drucker decodiert. Dies sind: LF (Line Feed), Papiervorschub um eine Zeile; FF (Form Feed), Papiervorschub zum nächsten Blatt; CR (Carriage Return), Ausdruck des Bufferspeichers und – wenn programmiert – anschließend Vorschub um eine Zeile; ESC (Escape), Umdefinieren der Zeichenbreite.

Der Zeichensatz umfaßt 96 ASCII-Symbole, die in einer 5 x 7-Punktmatrix mittels sieben übereinanderstehender Drucknadeln und eines Farbbandes gedruckt werden. Pro Zoll können 10, 12 oder 16,5 Zeichen auf Papier gebannt werden, wobei die gewünschte Zahl per Software angewählt werden kann.

Die Steuerung des gesamten Drukkers übernimmt ein eingebauter, kleiner 1-Chip-Mikrocomputer (MK 3870) und ein UART zur Serien-Parallel-Wandlung. Zur Steuerung der Motoren dienen Triacs als Schalter; die 20-mA-Schnittstelle ist mit einem Optokoppler realisiert. Ob noch genug Papier vorhanden ist und ob der Druckkopf in der Ruhelage steht, wird von zwei Lichtschranken registriert. Mechanik ist in einem Drucker zwar unumgänglich – hier wurde sie auf das vertretbare Minimum reduziert. Fe.

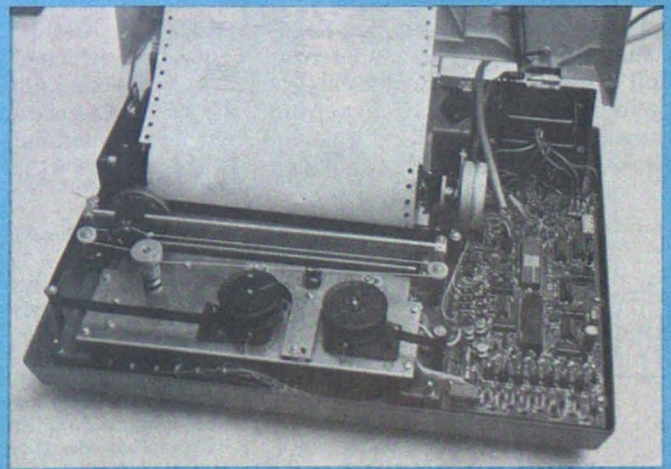


Bild 2. Auch für die Besitzer anderer Mikrocomputer-Systeme ist der Drucker H-14 interessant – vor allem wegen seines relativ geringen Bausatzpreises (Fotos: Knoll)

Impedanzuntersetzer-Berechnung

In seiner grundlegenden Form besteht die Schaltung eines Impedanzuntersetzers aus drei gleichartigen Impedanzen, von denen zwei feste Werte besitzen und die dritte variabel ist. *Bild 1* veranschaulicht dies anhand von Kapazitäten. C_1 und C_2 haben feste Werte, während C_T ein Trimmer ist. Die gezeigte Schaltung eignet sich dazu, den Bereich von C_T durch Zuschalten anderer Kapazitäten C_1 und C_2 zu verändern, wie dies z. B. für das Ziehen eines Quarzoszillators erforderlich wird, wenn Schaltungskapazitäten des Versuchaufbaues anders als erwartet sind und sich die Notwendigkeit ergibt, den Bereich des vorhandenen Trimmers zu verringern oder zu verlegen, um den Abgleich des Quarzoszillators bewerkstelligen zu können. Zahlenmäßig kann das so aussehen, daß ein Trimmer von 10...25 pF tatsächlich den Trimmerbereich C von 6 bis 7,5 pF haben sollte, der außerhalb des Einstellbereichs des Trimmers liegt. Mit dem in *Bild 2* gezeigten Programm für den Rechner SR-56 lassen sich die beiden Kapazitäten C_1 und C_2, die für diese Verlagerung des Trimmbereichs, wie in Bild 1 gezeigt, zugeschaltet werden müssen, leicht berechnen.

Definiert man mit C_o den oberen und mit C_u den unteren Wert des Trimmers C_T, mit C_3 den oberen und C_4 den unteren Wert des gewünschten Trimmbereichs von C, wären für das erstgenannte Beispiel $C_o = 25$ pF, $C_u = 10$ pF, $C_3 = 7{,}5$ pF und $C_4 = 6$ pF.

Die zu lösenden Gleichungen sind:

(1) $C_3C_0+C_3C_1+C_3C_2-C_0C_2-C_1C_2 = 0$

(2) $C_4C_u+C_4C_1+C_4C_2-C_uC_2-C_1C_2 = 0$

Sie entsprechen der Darstellung in Bild 1 und müssen für die beiden unbekannten Kapazitäten C_1 und C_2 gelöst werden. Da die beiden unbekannten Kapazitäten nichtlinear miteinander verknüpft sind und keine analytische Lösung für C_1 und C_2 zulassen, müssen die Lösungen iterativ ermittelt werden. Einfache iterative Verfahren, wenn verwendbar, liefern meist schon nach wenigen Iterationsschritten eine Lösung, haben aber den Nachteil, daß sie auf ihre Stabilität hin untersucht werden müssen. Für das vorliegende Problem kann ein einfaches Iterationsverfahren angewendet werden, wenn Gleichung (1) für C_2 und Gleichung (2) für C_1 gelöst wird.

Das beigefügte Rechnerprogramm ist denkbar einfach. Die bekannten Größen werden, wie im nachfolgenden Beispiel angegeben, in die entsprechenden Register eingegeben, worauf das Rechnerprogramm gestartet werden kann. Die Werte jeder Iteration werden kurz angezeigt: Der erste angezeigte Wert gibt den Wert für C_2, der zweite den für C_1 an. Der Wert für C_1 wird doppelt so lange angezeigt wie der von C_2. Auf diese Weise kann der Verlauf der Iterationen über die Anzeige leicht verfolgt werden. Ändern sich die Werte nicht mehr (nach 10 bis 20 Iterationen), kann der Rechengang durch Betätigen der R/S-Taste beendet werden.

Rechenbeispiel:
Es seien $C_o = 25$ pF, $C_u = 10$ pF, $C_3 = 7{,}5$ pF, $C_4 = 6$ pf. C_o wird in Register 1, C_u in Register 2, C_3 in Register 5 und C_4 in Register 6 geladen.

Nach Ablauf des Programms (nach etwa 20 Iterationen) enthält Register R3 den Wert für $C_1 = 5$ pf, R4 den Wert für $C_2 = 10$ pf. *Dipl.-Ing. Hans L. Steinmetz*

Video-Ausgabe beim AIM-65

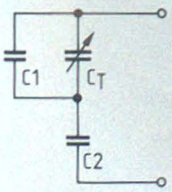
Der Mikrocomputer AIM-65 besitzt zwar ein 20stelliges alphanumerisches Display; beim Arbeiten mit dem Assembler oder in BASIC ist aber die Verwendung einer externen Video-Interface-Schaltung (z. B. VAB-2 von Mostek) recht praktisch. Leider besitzt das VAB-2 aber wiederum keine ASCII-Tastatur; es wäre daher nützlich, wenn das Video-Interface zusätzlich zum Display des AIM-65 mitliefe und dadurch die schon vorhandene AIM-Tastatur weiterbenützt werden könnte. Das Programm im *Bild* dient diesem Zweck. Bei der Inbetriebnahme ist wie folgt vorzugehen:

1. Das Programm im Bild laden (0170...01A4).
2. Um die Video-Ausgabe zu erreichen, schließt man das Interface an die serielle TTY-Schnittstelle des AIM an, schaltet es auf 300 Bd und startet das Programm an der Adresse 0170. Der TTY-Schalter auf der AIM-Platine bleibt dabei in Stellung KB.
3. Auf dem Bildschirm erscheint „Rockwell AIM-65". Alle Ein- und Ausgaben erfolgen jetzt parallel auf dem Video- und dem LED-Display.
4. Um die Ausgabe auf dem Bildschirm wieder zu unterbinden, startet man das Programm bei 0198. Das Video-Interface ist dann abgeschaltet; auf dem AIM-Display erscheint „Rockwell AIM-65".

Mit diesem Programm ist es möglich, ein Video-Interface oder einen Drucker mit serieller Schnittstelle zusammen mit der AIM-65-Tastatur zu betreiben

```
0170 A9 LDA #87
0172 8D STA A406
0175 A9 LDA #01
0177 8D STA A407
017A A9 LDA #0C
017C 8D STA A417
017F A9 LDA #C2
0181 8D STA A418
0184 6C JMP (FFFC)
0187 20 JSR EEA8
018A 29 AND #7F
018C C9 CMP #0D
018E D0 BNE 0197
0190 48 PHA
0191 A9 LDA #0A
0193 20 JSR EEA8
0196 68 PLA
0197 60 RTS
0198 A9 LDA #05
019A 8D STA A406
019D A9 LDA #EF
019F 8D STA A407
01A2 6C JMP (FFFC)
```

Selbstverständlich eignet sich die Routine auch für das Ausdrucken von Programmen auf einem Drucker mit serieller Schnittstelle. Die Baudrate kann an den Adressen 017B und 0180 nach den Angaben im AIM-Handbuch, auf den Seiten 9...29, geändert werden. Fe.



◀ Bild 1. Durch zwei zusätzliche Festkondensatoren läßt sich der Variationsbereich eines Drehkondensators oder Trimmers beliebig einengen; das funktioniert natürlich ebenso mit Leitwerten, d. h. mit Kehrwerten von Widerständen

Bild 2. ▶ Programm für den SR-56 von Texas Instruments. Es läßt sich ohne weiteres auch z. B. für den TI-58 oder TI-59 umschreiben

Adresse	Befehle
00	0 STO 3 RCL 5 x (RCL 1 + RCL 3) -
14	(RCL 1 + RCL 3 - RCL 5) =
25	STO 4 PAUSE RCL 6 x (RCL 2 + RCL 4)
38	- RCL 2 + RCL 4 = - (RCL 4 - RCL 6)
53	= STO 3 PAUSE PAUSE GTO 0 3

Hans-Georg Joepgen

„Copycheck" erschwert Programmdiebstahl

Verfahren zum Schutz urheberrechtlicher Vermerke

Seit geraumer Zeit vertreiben spezielle Software-Firmen Programme für gängige Personal Computer auf Kassetten. Ihr Wunsch, solche Programme vor unberechtigtem Kopieren und Auflisten zu schützen, ist nur verständlich.

Die Notwendigkeit eines Schutzverfahrens ergab sich aus Problemen mit dem Kartenspiel-Programm in [1]. Grundlagen bisheriger Schutzmethoden finden sich in [2]. An das zu entwikkelnde Verfahren waren folgende Anforderungen zu stellen: Rechengeschwindigkeit und Speicherplatzbedarf sollten nicht oder nur wenig zur ungünstigen Seite hin beeinflußt werden, bei unbefugter Entfernung der Copyright-Zeile sollte das Programm auf eine zuverlässige Weise unzuverlässig werden oder sich selbst zerstören. Schließlich war von dem Verfahren noch zu fordern, daß es sich bestenfalls nach aufwendiger Programm-Analyse durch größere Eingriffe außer Funktion setzen lassen durfte, und zum letzten war das Verfahren so auszugestalten, daß es – dies mit Rücksicht auf eine beabsichtigte Patentanmeldung – nicht auf einen einzigen Rechnertyp oder eine einzige Programmsprache beschränkt war.

Zur Erläuterung des Verfahrens ist eine kurze Betrachtung über die Art erforderlich, in der interpretierte Programmiersprachen Vermerke im Programm behandeln. Der Gesamt-Text eines Programms – Befehle, Daten, Zeilennummern und Vermerke – wird, beginnend mit einer maschinenspezifischen Adresse, im RAM-Bereich des Rechners abgelegt. Die Codierung der zum Programm gehörenden Befehlsworte erfolgt hierbei nach einem Schlüssel, der durch das Betriebssystem der betreffenden Maschine festgelegt ist und auf den der Programmierer keinen Einfluß hat. Stößt der Interpreter der Maschine bei der Abarbeitung eines Programmes auf einen Vermerk (eingeleitet durch REM in ANSI-Basic-Dialekten, durch C in Focal oder durch Sonderzeichen wie Strichpunkt, Komma und Apostroph in anderen höheren Programmiersprachen), so werden die nachfolgenden Bytes ignoriert. Sie erscheinen allein beim Listing-Ausdruck. Dennoch gibt es eine Möglichkeit, den Interpreter mittelbar auch hinter das Vermerk-Schlüsselwort blikken zu lassen, und zwar ist dies möglich über einen unmittelbaren Speicherzugriff mit Hilfe der Statements FMEM in Focal beziehungsweise PEEK in Basic. Beide Anweisungen verlangen die Angabe der absoluten Adresse im RAM-Bereich, aus der auszulesen ist.

An einer geeigneten Stelle im Programm werden nun stichprobenhaft RAM-Abfragen eingesetzt, wobei man auf jene Adressen zugreift, in denen der Text des Copyright-Vermerks gespeichert ist. Die Ergebnisse dieser Abfragungen werden auf eine beliebige Art (beispielsweise durch Subtraktion) miteinander verknüpft.

Das Resultat der Verknüpfung dient als Kriterium für einen bedingten Sprung: Nur dann, wenn das Verknüpfungsergebnis noch den erwarteten Wert zeigt und damit auf unveränderten Copyright-Text hinweist, wird beispielsweise eine Anweisung zum Löschen eines Programms übergangen.

In dieser Ausbaustufe arbeitet „Copycheck" bereits zufriedenstellend. Allerdings weist das Verfahren hier noch einen gewichtigen Mangel auf: Es ist vergleichsweise bequem dadurch außer Betrieb zu setzen, daß die Löschanweisung ihrerseits durch einen Fälscher aus dem Programm entfernt wird. Abhilfe bringt die Tatsache, daß nahezu jedes größere Programm neben Variablen auch ganzzahlige Konstanten erhält. So sind beispielsweise in der Basic-Zeile F=1/(2*PI*(SQR(L*C)) – der Thompsonschen Schwingungsformel, die der Errechnung der Frequenz F aus Kapazität C und Induktivität L dient – die Zahlen 1 und 2 solche ganzzahligen Konstanten. Weiter trifft man in Programmen Konstanten oft als Start- und Endwert von Schleifenzählern an, beispielsweise im Statement FOR N=1 TO 18. Geht man nun so vor, daß man diese Konstanten nicht im „Klartext" ins Programm einfügt, sondern statt ihrer das Ergebnis von Verknüpfungen der Speicherinhalte mit dem Copyright-Vermerk benutzt, dann werden Manipulationen am urheberrechtlichen Schutzvermerk zu fehlerhaftem Verhalten des Programms führen. Auch Radikalkuren, wie die Entfernung befremdlicher FMEMs oder PEEKs, helfen hier dem Programmdieb nicht weiter.

An einem konkreten Beispiel soll gezeigt werden, wie „Copycheck" arbeitet. Zur Demonstration dient der Euro-Apple ITT 2020, dessen nichtresidenter Applesoft-Interpreter den Text von Basic-Programmen ab Hexadresse 3001 aufbaut (Dezimal: 12289). Das Programm nach *Bild 1* bringt in den Zeilen 10 bis 37 erst einmal einen martialischen Urheberrechtsvermerk voller Säbelgerassel, es folgt – beispielhaft herausgearbeitet – das „Kamikaze"-Pro-

grammsegment mit der Selbstzerstörungsanweisung NEW. In Zeile 55 prüft das Programm stichprobenhaft drei Zeichen aus jenem Speicherbereich, in dem der Copyright-Vermerk liegt, setzt sie in Beziehung zueinander. Bei Nichtübereinstimmung druckt das Programm ein lakonisches VERFAELSCHT und verabschiedet sich sodann auf Nimmerwiedersehen. Ist der Vermerk laut Stichprobe unversehrt, erfolgt die Selbstlöschung nicht. Der Probelauf am Fuß des Rechnerausdrucks in Bild 1 zeigt's.

Was nun, wenn ein Bösewicht den Urhebervermerk listigerweise nahezu unverändert bestehen läßt und nur den Verfassernamen durch den eigenen austauscht? *Bild 2* zeigt, wie das Programm nun aussieht – und wie es sich benimmt. Bereits beim ersten Startversuch löst es sich in nichts auf!

Höhere Sicherheit gegen Verfälschungen des Autorenvermerks erzielt man durch die Ableitung von Konstantenwerten aus dem Copyrightbereich des Speichers. Hierfür bietet *Bild 3* ein Beispiel. In Zeile 40 werden zwei Buchstaben aus dem Autorennamen, der erste und dritte Buchstabe von Jonas, so mit einem internen Verwaltungszeichen aus der ersten Page des Basictextes verknüpft, daß als Startwert für die Schleife der Wert 3 gewonnen wird. Im Anschluß an das Listing findet sich der Prüflauf.

Bild 4 zeigt das gleiche Programm, diesmal allerdings dadurch verfälscht, daß Name und Anschrift des Urhebers verändert wurden. Die Reaktion des Programms auf die Manipulation zeigt hinwiederum der Prüflauf.

Wie wird Copycheck ins Programm eingebaut?

Bis jetzt wurde die Arbeitsweise des Copyright-Überprüfungsverfahrens Copycheck demonstriert, nicht jedoch, wie entsprechende Sicherungsroutinen ins Programm eingebaut werden. Die anfängliche Vermutung des Lesers, hierzu bedürfte es einer genauen Kenntnis der Arbeitsweise des eingesetzten Interpreters, trügt. Es reicht aus die Kenntnis jener RAM-Adresse, von der aus das Betriebssystem des Rechners den Basic-Text aufbaut. Man gibt als erste Programmzeilen (zeitlich zuerst und auch mit den kleinsten Zeilennummern) den Urheberschutzvermerk ein und druckt dann die ersten 256 Bytes des Speicherraums für den Basictext aus. Sodann wird das Programm gelöscht, und es folgt die Eingabe eines neuen Urheberrechtsvermerks – in Form von REM-Zeilen wiederum –, der an einigen charakteristischen Stellen von der ersten Fassung abweicht. Durch erneutes Ausdrucken des gleichen RAM-Bereiches wie zuvor wird man nun einige Bytes entdecken, die nicht in beiden Speicherausdrucken dekkungsgleich sind. Deren Werte und Adressen benutzt man nun (Werte aus Fassung eins bitte, dem Original-Copyrightvermerk), um mit ihrer Hilfe benötigte Konstanten zu gewinnen.

```
10  REM ------------------------
11  REM COPYRIGHT JOEPGEN, 1979
12  REM ------------------------
13  REM DUPLIZIEREN DIESES PRO-
14  REM GRAMMS IST NICHT ER-
15  REM LAUBT UND STELLT EINE
16  REM STRAFTAT DAR, DIE ZU
17  REM  + SCHADENERSATZ +
18  REM     ---------------
19  REM      VERPFLICHTET. -
20  REM FUER HINWEISE UEBER
21  REM MISSBRAUCH IST EINE
22  REM BELOHNUNG VON 1 000 DM
23  REM AUSGESETZT, DIE NACH
24  REM VERURTEILUNG DES TAETERS
25  REM UND NACH SEINER SCHADEN-
26  REM ERSATZLEISTUNG FAELLIG
27  REM          WIRD. -
28  REM RAUBKOPIEN SIND AM
29  REM FEHLEN DES ORIGINAL-
30  REM ETIKETTS AUF KASSETTE
31  REM ODER DISKETTE ERKENNBAR.
32  REM ENTFERNEN DIESES VER-
33  REM MERKS IST ALS URKUNDEN-
34  REM FAELSCHUNG STRAFBAR UND
35  REM FUEHRT ZUR ZERSTOERUNG
36  REM      DES PROGHRAMMS. -
37  REM -------------------------

50  REM
51  REM      DEMONSTRATION
52  REM
55  IF  PEEK (12333) -  PEEK (128
    20) <  >  PEEK (12798) THEN
     PRINT "VERFAELSCHT": NEW
60  PRINT "UNVERFAELSCHT": END

JRUN
UNVERFAELSCHT
```

Bild 1. Säbelgerassel im Copyright-Vermerk: So läuft das unverfälschte Programm

```
10  REM ------------------------
11  REM COPYRIGHT H.  GAUNER, 1979
12  REM ------------------------
13  REM DUPLIZIEREN DIESES PRO-
14  REM GRAMMS IST NICHT ER-
15  REM LAUBT UND STELLT EINE
16  REM STRAFTAT DAR, DIE ZU
17  REM  + SCHADENERSATZ +
18  REM     ---------------
19  REM      VERPFLICHTET. -
20  REM FUER HINWEISE UEBER
21  REM MISSBRAUCH IST EINE
22  REM BELOHNUNG VON 1 000 DM
23  REM AUSGESETZT, DIE NACH
24  REM VERURTEILUNG DES TAETERS
25  REM UND NACH SEINER SCHADEN-
26  REM ERSATZLEISTUNG FAELLIG
27  REM          WIRD. -
28  REM RAUBKOPIEN SIND AM
29  REM FEHLEN DES ORIGINAL-
30  REM ETIKETTS AUF KASSETTE
31  REM ODER DISKETTE ERKENNBAR.
32  REM ENTFERNEN DIESES VER-
33  REM MERKS IST ALS URKUNDEN-
34  REM FAELSCHUNG STRAFBAR UND
35  REM FUEHRT ZUR ZERSTOERUNG
36  REM      DES PROGHRAMMS. -
37  REM -------------------------

50  REM
51  REM      DEMONSTRATION
52  REM
55  IF  PEEK (12333) -  PEEK (128
    20) <  >  PEEK (12798) THEN
     PRINT "VERFAELSCHT": NEW
60  PRINT "UNVERFAELSCHT": END

JRUN
VERFAELSCHT
```

Bild 2. Hier hatte ein unehrlicher Zeitgenosse die Finger im Spiel: Das Programm reagiert mit Selbstlöschung auf Machenschaften am Copyright-Vermerk

```
2  REM ++++++++++++++++++++++++
3  REM +                      +
4  REM +   COPYCHECK-DEMO     +
5  REM +   ==============     +
6  REM +                      +
7  REM ++++++++++++++++++++++++
8  REM
10  REM ----------------------
12  REM COPYRIGHT: L.  ANGFINGER
14  REM VILLA LANDES
16  REM D 68 MANNHEIM
18  REM ----------------------
19  REM
20  REM  DEMONSTRATION
30  REM  -------------
35  PRINT "----------------------
    ----"
40  FOR N =  PEEK (12513) +  PEEK
    (12528) -  PEEK (12515) TO 5

50  PRINT : PRINT "ALLEIN DREIMAL
    IGES AUSDRUCKEN": PRINT "DIE
    SER ZEILE ZEIGT KORREKTE": PRINT
    "PROGRAMMAUSFUEHRUNG AN."
55  PRINT "----------------------
    ----"
60  NEXT : PRINT ".........": PRINT
    " +++ PROGRAMM BEENDET.": END

JRUN
-------------------------
ALLEIN DREIMALIGES AUSDRUCKEN
DIESER ZEILE ZEIGT KORREKTE
PROGRAMMAUSFUEHRUNG AN.
-------------------------
ALLEIN DREIMALIGES AUSDRUCKEN
DIESER ZEILE ZEIGT KORREKTE
PROGRAMMAUSFUEHRUNG AN.
-------------------------
ALLEIN DREIMALIGES AUSDRUCKEN
DIESER ZEILE ZEIGT KORREKTE
PROGRAMMAUSFUEHRUNG AN.
-------------------------
.........
 +++ PROGRAMM BEENDET.
```

Bild 3. Copycheck-Demonstration mit Konstantengewinnung: Herrn Redlich folgt der Rechner aufs Wort

```
2  REM ++++++++++++++++++++++++
3  REM +                      +
4  REM +   COPYCHECK-DEMO     +
5  REM +   ==============     +
6  REM +                      +
7  REM ++++++++++++++++++++++++
8  REM
10  REM ------------------------
12  REM COPYRIGHT JONAS REDLICH
14  REM DICHTERSTRASSE 17
16  REM D 4812 BORGHAUSEN
18  REM ------------------------
19  REM
20  REM  DEMONSTRATION
30  REM  -------------
35  PRINT "------------------------
    ----"
40  FOR N =  PEEK (12513) +  PEEK
    (12528) -  PEEK (12515) TO 5

50  PRINT : PRINT "ALLEIN DREIMAL
    IGES AUSDRUCKEN": PRINT "DIE
    SER ZEILE ZEIGT KORREKTE": PRINT
    "PROGRAMMAUSFUEHRUNG AN. "
55  PRINT "------------------------
    ----"
60  NEXT : PRINT ".........": PRINT
    " +++ PROGRAMM BEENDET. ": END
]RUN
--------------------------
ALLEIN DREIMALIGES AUSDRUCKEN
DIESER ZEILE ZEIGT KORREKTE
PROGRAMMAUSFUEHRUNG AN.
--------------------------
 +++ PROGRAMM BEENDET.
```

Bild 4. Wieder war ein Dunkelmann am Werk und fälschte sich unverdienten Lorbeer in den Copyright-Vermerk. Nun löscht sich das Programm nicht mehr, sondern straft mit falschen Ergebnissen

Bild 5 zeigt, in Hex-Darstellung, den zum Programm von Bild 3 gehörenden Speicherausdruck und den Speicherausdruck des verfälschten Programms nach Bild 4. Copycheck verwendet hier die Bytes in den Speicherzellen 30E3, 30E1 und 30F0 (Dezimal 12515, 12513 und 12528). Sofern das Betriebssystem des Rechners nicht unmittelbar die Möglichkeit zum Hex-Druck von Speicherinhalten auf Schirm oder Papier bietet – so fehlt zum Beispiel dem PET 2001 dieses Talent – kann man sich durch eine PEEK-Schleife in Immediate-Mode behelfen; hierbei erfolgt der Ausdruck dezimal, was jedoch eher eine Erleichterung denn eine Erschwerung bei der Suche nach geeigneten Kontrollbytes bedeutet.

Wie die Verknüpfung der gefundenen Bytes im einzelnen geschieht, ist der Fantasie des Programmierers überlassen – je trickreicher Copycheck im Programm versteckt ist, desto mühsamer wird sich das Geschäft Programmanalyse und Programmverfälschung für unehrliche Zeitgenossen gestalten, die sich mit fremden Federn zu schmücken wünschen oder Programmierer um jene vergleichsweise geringen Beträge prellen möchten, die von gewerblichen Programmverkäufern üblicherweise für den Autor aus dem Erlös abgezweigt werden.

Literatur

[1] *Hans-Georg Joepgen*: „Mit Computerhilfe ‚zurück in die Vergangenheit' – PET als listenreicher Kartenspieler". Hobbycomputer 1, Sonderheft des Franzis-Verlags, München.

[2] *Len Lindsay*: „List Protection" in „Kilobaud Microcomputing", Juni-Heft 1979, Verlag 1001001 Incorporated, Peterborrough, USA – Europa-Vertrieb „Kilobaud", 7778 Markdorf.

Berichtigung

TI-59 druckt Kalender

FUNKSCHAU 1979, Heft 22, Seite 1312

Beim Abdruck des Programms wurden leider versehentlich die Inhalte der Speicherregister nicht mit angegeben, obwohl sie (als Block 4) ebenfalls erforderlich und nun nachträglich dem untenstehenden *Bild* zu entnehmen sind. Und vergessen Sie nicht, die Speicherbereichsverteilung mit 5 OP 17 zu ändern!

```
*3000.3100
3000- 00 1E 30 02 00 B2 2B 2B
3008- 2B 2B 2B 2B 2B 2B 2B 2B
3010- 2B 2B 2B 2B 2B 2B 2B 2B
3018- 2B 2B 2B 2B 2B 00 3B 30
3020- 03 00 B2 2B 20 20 20 20
3028- 20 20 20 20 20 20 20 20
3030- 20 20 20 20 20 20 20 20
3038- 20 2B 00 58 30 04 00 B2
3040- 2B 20 20 20 43 4F 50 59
3048- 43 48 45 43 4B 2D 44 45
3050- 4D 4F 20 20 20 20 2B 00
3058- 75 30 05 00 B2 2B 20 20
3060- 20 3D 3D 3D 3D 3D 3D 3D
3068- 3D 3D 3D 3D 3D 3D 3D 20
3070- 20 20 20 2B 00 92 30 06
3078- 00 B2 2B 20 20 20 20 20
3080- 20 20 20 20 20 20 20 20
3088- 20 20 20 20 20 20 20 20
3090- 2B 00 AF 30 07 00 B2 2B
3098- 2B 2B 2B 2B 2B 2B 2B 2B
30A0- 2B 2B 2B 2B 2B 2B 2B 2B
30A8- 2B 2B 2B 2B 2B 2B 00 B5
30B0- 30 08 00 B2 00 D2 30 0A
30B8- 00 B2 2D 2D 2D 2D 2D 2D
30C0- 2D 2D 2D 2D 2D 2D 2D 2D
30C8- 2D 2D 2D 2D 2D 2D 2D 2D
30D0- 2D 00 F0 30 0C 00 B2 43
30D8- 4F 50 59 52 49 47 48 54
30E0- 20 4A 4F 4E 41 53 20 52
30E8- 45 44 4C 49 43 48 20 00
30F0- 07 31 0E 00 B2 44 49 43
30F8- 48 54 45 52 53 54 52 41
3100- 53
```

```
*3000.3100
3000- 00 1E 30 02 00 B2 2B 2B
3008- 2B 2B 2B 2B 2B 2B 2B 2B
3010- 2B 2B 2B 2B 2B 2B 2B 2B
3018- 2B 2B 2B 2B 2B 00 3B 30
3020- 03 00 B2 2B 20 20 20 20
3028- 20 20 20 20 20 20 20 20
3030- 20 20 20 20 20 20 20 20
3038- 20 2B 00 58 30 04 00 B2
3040- 2B 20 20 20 43 4F 50 59
3048- 43 48 45 43 4B 2D 44 45
3050- 4D 4F 20 20 20 20 2B 00
3058- 75 30 05 00 B2 2B 20 20
3060- 20 3D 3D 3D 3D 3D 3D 3D
3068- 3D 3D 3D 3D 3D 3D 3D 20
3070- 20 20 20 2B 00 92 30 06
3078- 00 B2 2B 20 20 20 20 20
3080- 20 20 20 20 20 20 20 20
3088- 20 20 20 20 20 20 20 20
3090- 2B 00 AF 30 07 00 B2 2B
3098- 2B 2B 2B 2B 2B 2B 2B 2B
30A0- 2B 2B 2B 2B 2B 2B 2B 2B
30A8- 2B 2B 2B 2B 2B 2B 00 B5
30B0- 30 08 00 B2 00 D2 30 0A
30B8- 00 B2 2D 2D 2D 2D 2D 2D
30C0- 2D 2D 2D 2D 2D 2D 2D 2D
30C8- 2D 2D 2D 2D 2D 2D 2D 2D
30D0- 2D 00 EF 30 0C 00 B2 43
30D8- 4F 50 59 52 49 47 48 54
30E0- 3A 20 4C 2E 20 41 4E 47
30E8- 46 49 4E 47 45 52 00 01
30F0- 31 0E 00 B2 56 49 4C 4C
30F8- 41 20 4C 41 4E 44 45 53
3100- 00
```

Bild 5. Die Gegenüberstellung von Speicherausdrucken mit den Basic-Texten von Original-Autorenvermerk und abgeänderter Version zeigt Adressen und Inhalte jener Speicherzellen, die sich für Copycheck zur Kontrolle anbieten. Intimere Kenntnis des Interpreters ist beim Programmierer nicht vonnöten – links das Programm des Herrn Redlich, rechts die Fälschung

```
2500130031.      15
  41001300.      16
3500000000.      17
2100170014.      18
  35004100.      19
1300350000.      20
3000130017.      21
  35004600.      22
1300330035.      23
  24002700.      24
3000130024.      25
2500410031.      26
  24000000.      27
2500410027.      28
  24000000.      29
1300410022.      30
  41003600.      31
3700000000.      32
3600170033.      33
  37001700.      34
3000140017.      35
  35000000.      36
3200260037.      37
  32001400.      38
1700350000.      39
3100320042.      40
  17003000.      41
1400170035.      42
1600170046.      43
  17003000.      44
1400170035.      45
3032001624.      46
  30240016.      47
3200213500.      48
3613003632.      49
```

Funkschau Mikro-Poster

Unser Leser Hans Lauer hat sich die Mühe gemacht, alle hexadezimalen 8080-Operationscodes in einer übersichtlichen Tabelle zusammenzustellen und mit einer kurzen Erklärung zu versehen. Wir sind sicher, daß für 8080-Anwender das Programmieren in Maschinensprache damit wesentlich einfacher und zeitsparender wird.

8080-Operationscodes

BEFEHLSLISTE FUER DEN MIKROPROZESSOR 8080A/INTEL

TRANSFERBEFEHLE

	A	B	C	D	E	H	L	RPH*	BYTE	
MOV A	7F	78	79	7A	7B	7C	7D	7E	1	LADE A MIT INHALT V. REGISTER ..
MOV B	47	40	41	42	43	44	45	46	1	" B " " "
MOV C	4F	48	49	4A	4B	4C	4D	4E	1	" C " " "
MOV D	57	50	51	52	53	54	55	56	1	" D " " "
MOV E	5F	58	59	5A	5B	5C	5D	5E	1	" E " " "
MOV H	67	60	61	62	63	64	65	66	1	" H " " "
MOV L	6F	68	69	6A	6B	6C	6D	6E	1	" L " " "
MOV RPH	77	70	71	72	73	74	75	--	1	LADE DAT. IN D. ADR. D. IM RPH STEHT

* MOV R, RPH: LADE REG. MIT DEN DATEN DEREN ADRESSE IM RPH ANGEGEBEN IST

OPERATIONEN AKKU - REGISTER

	A	B	C	D	E	H	L	RPH*	BYTE	
MVI	3E	06	0E	16	1E	26	2E	36	2	LADE REG. MIT DER KONSTANTEN
INR	3C	04	0C	14	1C	24	2C	34	1	INHALT DES REGISTERS PLUS 1
DCR	3D	05	0D	15	1D	25	2D	35	1	INHALT DES REGISTERS MINUS 1
ADD	87	80	81	82	83	84	85	86	1	ADDIERE INHALT DES REG. ZUM AKKU
SUB	97	90	91	92	93	94	95	96	1	SUBTRAH. INHALT DES REG. VOM AKKU
ADC	8F	88	89	8A	8B	8C	8D	8E	1	AKKU PLUS REG. PLUS UEBERTRAG
SBB	9F	98	99	9A	9B	9C	9D	9E	1	AKKU MINUS REG. MINUS UEBERTRAG
ANA	A7	A0	A1	A2	A3	A4	A5	A6	1	AKKU UND REG. WERDEN UND VERKN.

SPRUNGBEFEHLE

	JUMP	CALL	RETURN	BYTE	
UNBED.	C3	CD	C9	3/3/1	UNBED. SPRUNGAR, JMP / CALL / RET
* NZ	C2	C4	C0	3/3/1	AUSFUEHR. NUR WENN Z=0 (NU)
* Z	CA	CC	C8	3/3/1	AUSFUEHR. NUR WENN Z=1 (NU)
* NC	D2	D4	D0	3/3/1	AUSFUEHR. NUR WENN CY=0 (UEB)
* C	DA	DC	D8	3/3/1	AUSFUEHR. NUR WENN CY=1 (UEB)
* PO	E2	E4	E0	3/3/1	AUSFUEHR. NUR WENN P=0 (GER)
* PE	EA	EC	E8	3/3/1	AUSFUEHR. NUR WENN P=1 (GER)
* P	F2	F4	F0	3/3/1	AUSFUEHR. NUR WENN S=0 (NE)
* M	FA	FC	F8	3/3/1	AUSFUEHR. NUR WENN S=1 (NE)

* = J ODER C ODER R, C.. = UNTERPR. -AUFR. R.. = RUECKSP. -BEFEHL
BEI ALLEN AUFRUFBEFEHLEN WIRD DIE RUECKKEHRADRESSE IM SP ABGELEGT
BEI A. RUCKSP. -BEF. WIRD DAS PROGR. AN DER ADRESSE FORTGESETZT WELCHE
DER INHALT DER ADRESSE IST DIE IM STAPELZEIGER ANGEGEBEN IST

RESTART-BEFEHLE

		BYTE	
RST 0	C7	1	PROGRAMM WIRD BEI DER ADRESSE 0 FORTGESETZT
RST 1	CF	1	PROGRAMM WIRD BEI DER ADRESSE 8 FORTGESETZT
RST 2	D7	1	PROGRAMM WIRD BEI DER ADRESSE 16DEZ FORTGESETZT
RST 3	DF	1	PROGRAMM WIRD BEI DER ADRESSE 24DEZ FORTGESETZT
RST 4	E7	1	PROGRAMM WIRD BEI DER ADRESSE 32DEZ FORTGESETZT

ORA		B7	B0	B1	B2	B3	B4	B5	B6	1	AKKU UND REG. WERDEN ODER VERKN.
XRA		AF	A8	A9	AA	AB	AC	AD	AE	1	AKKU UND REG. WERDEN EXOR VERKN.
CMP		BF	B8	B9	BA	BB	BC	BD	BE	1	AKKU WIRD MIT REG. VERGLICHEN

* IM RPH STEHT DIE ADRESSE MIT DEREN DATEN DIE OPERATION AUSGEF. WIRD

SPEZIELLE OPERATIONEN (NUR MIT AKKU)

		BEEINF. Z	FLAGS S	P	C	A	BYTE	
ADI	C6	*	*	*	*	*	2	ADDIERE KONSTANTE ZUM INHALT DES AKKUS
ACI	CE	*	*	*	*	*	2	ADD. KONST. UND CARRY ZUM INHALT D. AKKUS
SUI	D6	*	*	*	*	*	2	SUBTR. KONSTANTE VOM INHALT DES AKKUS
SBI	DE	*	*	*	*	*	2	SUB. KONST. UND CARRY VOM INHALT D. AKKUS
CMA	2F	-	-	-	-	-	1	AKKUINHALT WIRD NEGIERT
ANI	E6	*	*	*	0	1	2	AKKU WIRD MIT D. KONSTANTEN UND VERKN.
ORI	F6	*	*	*	0	0	2	AKKU WIRD MIT D. KONSTANTEN ODER VERKN.
XRI	EE	*	*	*	0	0	2	AKKU WIRD MIT D. KONSTANTEN EXOR VERKN.
CPI	FE	*	*	*	*	*	2	AKKU WIRD MIT D. KONSTANTEN VERGLICHEN
DAA	27	*	*	*	*	*	1	DEZIMALKORREKTUR EINES WERTES IM AKKU

SPEZIELLE TRANSFERS VON / NACH AKKU, KEINE BEEINFLUSSUNG DER FLAGS !

		BYTE	
LDA	3A	3	LADE AKKU MIT DEM INHALT DER SPEICHERSTELLE
STA	32	3	SPEICHERE INHALT DES AKKUS IN DER SPEICHERSTELLE
LDAX B	0A	1	LADE AKKU MIT DEN DATEN DERN ADRESSE IM RP(B,C) STEHT
STAX B	02	1	LADE ADR. DIE IM RP(B,C) STEHT MIT DEN DATEN DES AKKUS
LDAX D	1A	1	LADE AKKU MIT DEN DATEN DERN ADRESSE IM RP(D,E) STEHT
STAX D	12	1	LADE ADR. DIE IM RP(D,E) STEHT MIT DEN DATEN DES AKKUS
IN	DB	2	AKKU WIRD MIT DEM INHALT DES EINGABEKANALS GELADEN
OUT	D3	2	INHALT DES AKKUS WIRD AUF DEN AUSGABEKANAL GEGEBEN

DOPPELREGISTER OPERATIONEN

	B,C	D,E	H,L	SP*	A/F	BYTE	
LXI	01	11	21	31	-	3	LADE REGPAAR MIT DER KONSTANT
DAD	09	19	29	39	-	1	ADD. INHALT D. REGP. Z. INHALT RPH
INX	03	13	23	33	-	1	INHALT D. REGPAARS PLUS 1
DCX	0B	1B	2B	3B	-	1	INHALT D. REGPAARS MINUS 1
PUSH	C5	D5	E5	--	F5	1	INH. D. REP. IN ADR D. IM SP STEHT
POP	C1	D1	E1	--	F1	1	INHALT DES SP IN REGPAAR LADEN

* DIE OPERATION WIRD MIT DEM INHALT DES STAPELZEIGERS AUSGEFUEHRT

SPEZIELLE TRANSFERS VON REGISTERPAAREN, KEINE BEEINFLUSSUNG DER FLAGS !

		BYTE	
LHLD	2A	3	LADE RPH MIT D. INH. DER ANGEGEBENEN ADRESSE U. (ADR+1)
SHLD	22	3	INH. DES RPH UNTER D. ANGEG. ADRESSE U. (ADR+1) SPEICHERN
XCHG	EB	1	VERTAUSCHE DEN INHALT DER REGP. (D,E) UND REGP. (H,L)
XTHL	E3	1	VERT. DEN INH. DES RPH MIT D. INH. D. ADR. DIE D. SP ADR. IST
SPHL	F9	1	LADE STAPELZEIGER MIT DEM INH. DES REGISTERPAARS (H,L)
PCHL	E9	1	PROGRAMM WIRD AN DER ADR. FORTGESETZT DIE IM RPH STEHT
RST 5	EF	1	PROGRAMM WIRD BEI DER ADRESSE 40DEZ FORTGESETZT
RST 6	F7	1	PROGRAMM WIRD BEI DER ADRESSE 48DEZ FORTGESETZT
RST 7	FF	1	PROGRAMM WIRD BEI DER ADRESSE 56DEZ FORTGESETZT

STEUERBEFEHLE

		BYTE	
EI	FB	1	INTERRUPT-FF. WIRD GESETZT CPU KANN UNTERBR. ANNEHMEN
DI	F3	1	INTERRUPT-FF. WIRD RUCKGES. CPU IGNORIERT UNTERBR. -ANF.
STC	37	1	CARRY-BIT WIRD GESETZT
CMC	3F	1	CARRY-BIT WIRD NEGIERT
HLT	76	1	PROGR. HAELT AN, BIS EINE UNTERB. -ANFORDERUNG EINTRIFFT

AKKUMULATOR-ROTIEREN (VERSCHIEBEN IM AKKU)

		BYTE	
RLC	07	1	AKKU-INH. UM 1BIT N. LINKS BIT 2\7 N. CY-B.
RAL	17	1	AKKU-INH. UM 1BIT N. LINKS BIT 2\7 N. CY-B. CY-B. N. 2\0
RRC	0F	1	AKKU-INH. UM 1BIT N. RECHTS BIT 2\0 N. CY-B.
RAR	1F	1	AKKU-INH. UM 1BIT N. RECHTS BIT 2\0 N. CY-B. CY-B. N. 2\7

LETZTER BEFEHL !!

		BYTE	
NOP	00	1	NO OPERATION, LEERBEFEHL

BEEINFLUSSUNG DER FLAGS BEIM 8080A/INTEL

	BEEINFL. Z	FLAGS S	P	C	A		
ACI	*	*	*	*	*		
ADC	*	*	*	*	*	Z (ZERO-FLAG)	= NU (NULLZUSTAND)
ADD	*	*	*	*	*	S (SIGN-FLAG)	= NE (VORZEICHEN-FLAG)
ADI	*	*	*	*	*	P (PARITY-FLAG)	= GE (GERADHEITSZUST.)
ANA	*	*	*	*	*	CY (CARRY-FLAG)	= UEB (UEBERTRAGSZUST.)
ANI	*	*	*	0	1	AC (AUXILIARY-CARRY-FLAG)	= HUE (HILFS UEBERTRAG)
CMC	-	-	-	*	-		
CMP	*	*	*	*	*		
CPI	*	*	*	*	*		
DAA	*	*	*	*	*		
DAD	-	-	-	*	-		
DCR	*	*	*	-	*		
INR	*	*	*	-	*		
ORA	*	*	*	*	*		
ORI	*	*	*	0	0		
POP A/F	*	*	*	*	*	(A/F) = PSW	
RAL	-	-	-	*	-		
RAR	-	-	-	*	-		
RLC	-	-	-	*	-		
RRC	-	-	-	*	-		
SBB	*	*	*	*	*		
SBI	*	*	*	*	*		
STC	-	-	-	*	-		
SUB	*	*	*	*	*		
SUI	*	*	*	*	*		
XRA	*	*	*	*	*		
XRI	*	*	*	0	0		

ZUORDNUNG DER FLAGBITS IM FLAGREGISTER:

2\7	2\6	2\5	2\4	2\3	2\2	2\1	2\0
S	Z	0	A	0	P	1	C

ALLE ANDEREN BEFEHLSTYPEN HABEN KEINEN EINFLUSS AUF DIE FLAGS

Komfort-Platine

Der Mikrocomputer „Eurocom-1"

Einer der preiswertesten Einplatinen-Computer auf der Basis der CPU 6802 von Motorola ist derzeit der „Eurocom-1", der von der Mainzer Firma Eltec gefertigt und vertrieben wird, die vor einiger Zeit das SCMP-Billigsystem MK 14 vertrieb (vgl. FUNKSCHAU 1978,Heft 21). Der Eurocom-1 gestattet in der Grundversion die Eingabe hexadezimaler Befehle und Daten mit einer Hex-Tastatur. Zur Aufzeichnung von Programmen dient ein Kassetten-Interface auf der Platine, das nach dem Kansas-City-Standard und mit einem Motorola-kompatiblen Datenformat arbeitet. Ferner enthält die Karte zwei PIAs mit zusammen 40 programmierbaren Ein- und Ausgabeleitungen und 1 KByte RAM. Die CPU 6802 ist mit der 6800 software-kompatibel.

Das Eurocom-Betriebssystem

Das Monitorprogramm ist fest in zwei 2708-EPROMs gespeichert und belegt 2 KByte. Die Verwendung der EPROMs bedingt die Speisespannungen + 5 V, −5 V und + 12 V; ein passendes Netzgerät ist lieferbar. Das Monitorprogramm ist deutlich komfortabler ausgelegt als z.B. das des KIM-1. Es gestattet das Setzen von max. 15 Breakpoints, den Einzelschritt-Betrieb, das Anzeigen der CPU-Register, die Errechnung relativer Branch-Adressen und das Aus- und Einlesen von Programmen mittels des Kassettenrecorders bei 300 bit/s.

Zum Anschluß eines ASCII-Terminals sind EPROMs mit entsprechend geändertem Betriebssystem erhältlich, ferner auch RAM-Erweiterungskarten und ein Video-Interface. Das System ist bis zum Basic-Rechner ausbaubar; auch ein 6802-Assembler steht zur Verfügung.

Die mitgelieferten Unterlagen sind deutschsprachig und enthalten, um dem Benutzer erste Erfolgserlebnisse nach dem Einschalten zu ermöglichen, zehn einfache Beispielprogramme, z.B. zur Verwendung des Eurocom als Digitaluhr mit Wecker. Außerdem findet man darin Datenblätter aller ICs auf der Platine und ein gut kommentiertes Listing des Monitorprogrammes.

Der mechanische Aufbau...

Der Mikrocomputer Eurocom-1 ist auf einer zweiseitig kaschierten Epoxy-Leiterplatte im doppelten Europakartenformat aufgebaut; Schaltkreise wie Mikroprozessor, EPROMs, ROM und RAMs sitzen auf Sockeln. Das Board wird über eine 64polige Steckerleiste kontaktiert, abgesehen von den Lötstiften für den Kassettenrecorder. Als Display ist oberhalb der Tastatur eine 8 mm hohe, achtstellige Siebensegment-LED-Anzeige vorhanden. 30 Einzeltasten bilden die Tastatur, die eine hexadezimale Eingabe und teilweise eine direkte Eingabe von Befehlen erlaubt. Die Bezeichnung der Tasten ist nicht graviert, sondern mit einer Klebefolie aufgebracht. Ein Teil der Leiterplatte ist mit einem Lochraster versehen, das eine nachträgliche, individuelle Bestückung des Boards, z.B. mit A/D-Wandlern, ermöglicht; ebenso kann eine zweite 64polige Stiftleiste angebracht werden.

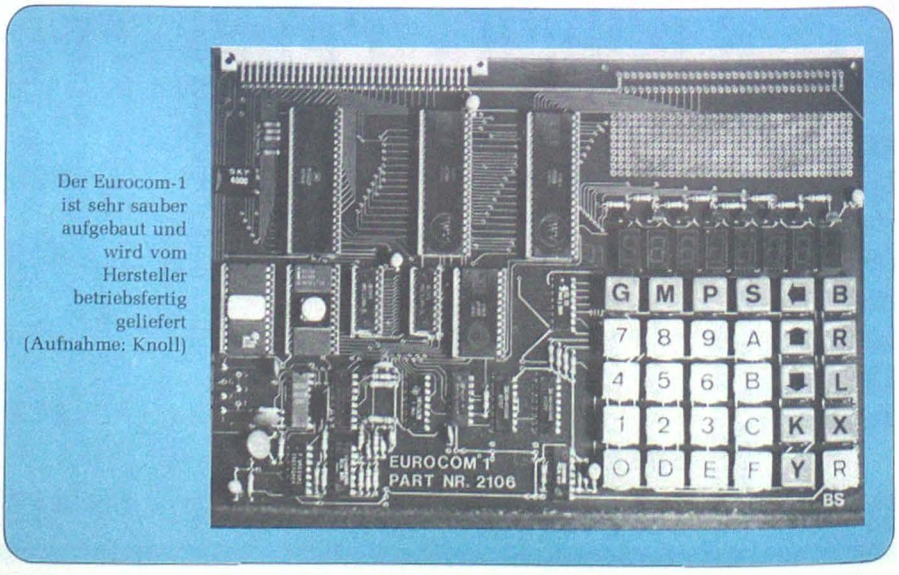
...und die Erfahrungen damit

Der Mikrocomputer ist sehr sauber aufgebaut und dadurch, daß er bereits betriebsfertig geliefert wird, auch für Nichtelektroniker geeignet. Der einzige Mangel ist die Tastatur. Auf der Tastenfläche ist dort, wo man sonst eine saubere Gravur vorfindet, ein Schild aufgeklebt, das sich bereits nach kurzem Gebrauch ablöst. Auch vom Hersteller wurde das Manko erkannt und für die nächsten Lieferungen Abhilfe in Form von gravierten Tastenköpfen versprochen.

Diese zukünftige Verbesserung kann jedoch nicht darüber hinwegtäuschen, daß die Tasten grundsätzlich für diesen Einsatz nicht geeignet sind. Bereits nach kurzer Zeit stellte sich gelegentliches Prellen ein, das sich bei der Eingabe langer Programme als sehr störend erwies. Doch auch an diesem Punkt wird der Hersteller noch eine andere Lösung finden und dann einen Mikrocomputer anbieten, der in dieser Preisklasse seinesgleichen sucht. Fe/Kn

Adressenbelegung des Eurocom-1

0000...007F RAM in der CPU 6802
8004...8007 Anwender-PIA-Adressen
8008...8009 Kassetten-ACIA-Interface
8010...8013 System-PIA (Anzeige usw.)
A400...A7FF RAM; frei bis ca. A7A0
F800...FFFF EPROMs m. Monitorprogramm



Der Eurocom-1 ist sehr sauber aufgebaut und wird vom Hersteller betriebsfertig geliefert (Aufnahme: Knoll)

Jörg Lohse

KIM-Monitorprogramm für Baudot-Fernschreiber

Die Bedienung eines Mikrocomputers über einen Fernschreiber ist aus verschiedenen Gründen wünschenswert. Bei der Eingabe von Programmen erhält die Bedienungsperson ein schriftliches Protokoll, das bei der Beseitigung von Fehlern im Programm eine große Hilfe darstellt. Die größere, schreibmaschinenähnliche Tastatur eines Fernschreibers ist bequemer und ermüdungsfreier zu bedienen als die oftmals gedrängt zusammenliegenden Tasten an Mikrocomputern.

Bei der Auswahl von Fernschreibern gibt es grundsätzlich folgende zwei Möglichkeiten: Fernschreiber, die den ASCII-Code benutzen, und solche, die sich des Baudot-Codes bedienen. Die ASCII-Fernschreiber verfügen über Groß- und Kleinbuchstaben sowie Sonderzeichen. Der KIM-1 ist, wie viele andere Mikrocomputer, in der Lage, Zeichen im ASCII-Code zu senden und zu empfangen. ASCII-Fernschreiber haben allerdings den großen Nachteil, daß sie teuer sind.

Baudot-Fernschreiber beherrschen außer Ziffern und Zeichen nur Groß- oder Kleinbuchstaben. Der Code ist in Mikrocomputern nicht gebräuchlich; er ist komplizierter als der ASCII-Code, denn es müssen zwischen Buchstaben und Ziffern oder Zeichen Umschaltsignale gegeben werden. Aus dem öffentlichen Telexnetz stammende Baudot-Fernschreiber sind allerdings geprüft und instandgesetzt preiswert erhältlich. Sie kosten oft weniger als „Billigdrucker" für Mikrocomputer und verfügen im Gegensatz zu jenen über eine Tastatur und über direkten Netzanschluß.

Das folgende Programm erlaubt die Bedienung des KIM-1 mittels eines Baudot-Fernschreibers. Der Verfasser benutzte den Fernschreiber Siemens T 37, der häufig angeboten wird und die Benutzung von DIN-A-4-Papier erlaubt. Die Benutzung ähnlicher Fernschreiber (Lorenz LO-15, Siemens T-100 oder T-68) ist ebenfalls möglich.

Anschluß des Fernschreibers

Der Anschluß des Fernschreibers an den KIM-1 wird in *Bild 1* gezeigt. Das Programm benutzt zur Kommunikation die ASCII-Ein- und Ausgänge des KIM. T1 puffert den Ausgang des Mikrocomputers, da der Fernschreiber 40 mA Linienstrom benötigt. Die Dioden schützen die Logik auf der KIM-Platine und T1 vor Induktionsspitzen vom Empfangsmagneten. Da eine zum Empfangsmagneten antiparallel geschaltete Diode die Schaltzeiten verlängern würde, dienen die beiden hochspannungsfesten Dioden dem Schutz vor Induktionsspitzen.

Der Sendekontakt kann ohne besondere Maßnahmen mit dem Interface auf der KIM-Platine verbunden werden. Die Angabe der Farbe bei den Anschlußdrähten des Fernschreibers bezieht sich auf den Siemens T 37. Wenn ein anderer Typ zur Anwendung kommen soll, kann der Empfangsmagnet durch seinen Widerstand von 50...200 Ω ausfindig gemacht werden.

Das Fernschreiberinterface arbeitet auch, wenn das Monitorprogramm nicht läuft. Das Interface kann also vor der Eingabe des Programms geprüft

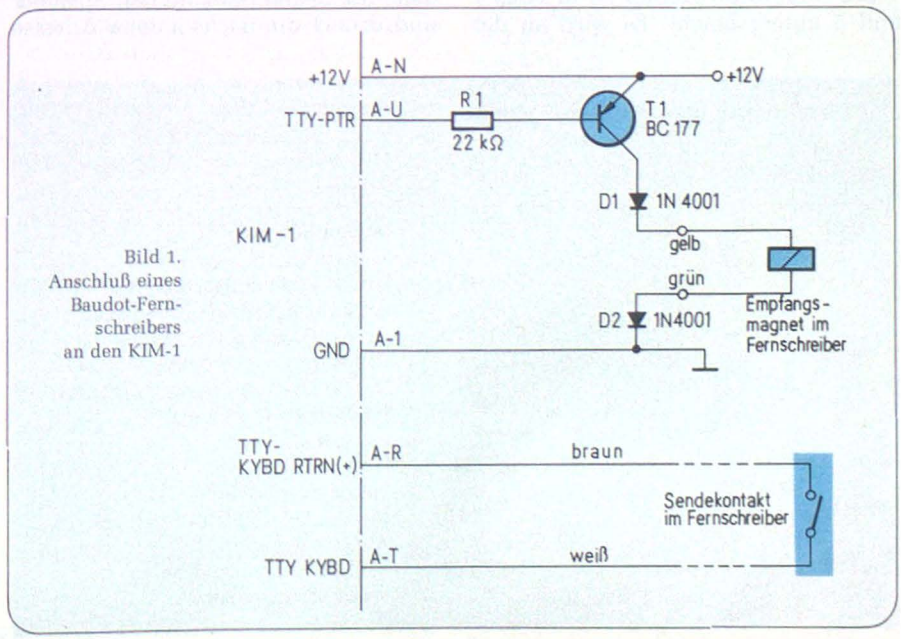


Bild 1. Anschluß eines Baudot-Fernschreibers an den KIM-1

werden: 5 V und 12 V werden an den KIM-1 geschaltet, und die Reset-Taste wird gedrückt. Dann kann der Fernschreiber mit der Netzspannung verbunden werden. Das Fernschreiberinterface erlaubt durch eine logische Verknüpfung auf der KIM-Platine, daß die auf der Tastatur bestätigten Buchstaben vom Drucker ausgegeben werden. Wenn mit der Tastatur auf dem Drucker geschrieben werden kann, haben Fernschreiber und Interface ihre Prüfung bestanden, und das Programm kann eingegeben und gestartet werden.

```
0400 20 8C 1E 20 43 05 A9 4B 20 52 05 A9 49 20 52 05
0410 A9 4D 20 52 05 20 43 05 20 13 05 20 50 05 20 0C
0420 05 20 50 05 A9 00 85 F8 85 F9 20 8A 04 20 AC 1F
0430 C9 20 D0 06 20 CC 1F 4C 15 04 C9 4D D0 03 4C 4F
0440 1C C9 0D D0 06 20 63 1F 4C 15 04 C9 0A D0 0E 38
0450 A5 FA E9 01 85 FA B0 02 C6 FB 4C 15 04 C9 2E D0
0460 0C A0 00 A5 F8 91 FA 20 63 1F 4C 15 04 C9 47 D0
0470 03 4C C8 1D 4C 2A 04 20 8A 04 20 AC 1F 20 8A 04
0480 20 AC 1F A5 F8 60 EA EA EA EA 86 FD A9 80 2C 40
0490 17 D0 FB A2 00 86 FE 20 AE 05 A0 05 AD 40 17 29
04A0 80 46 FE 05 FE 85 FE 20 B4 05 88 D0 EF 46 FE 46
04B0 FE 46 FE 20 B4 05 A5 FE C9 1B D0 07 A9 20 85 EA
04C0 4C 8A 04 C9 1F D0 07 A9 00 85 EA 4C 8A 04 C9 02
04D0 D0 05 A9 0A 4C 07 05 C9 08 D0 05 A9 0D 4C 07 05
04E0 C9 04 D0 05 A9 20 4C 07 05 EA 05 EA 85 FE A2 FF
04F0 E8 BD C0 05 C5 FE D0 F8 86 FE A9 20 24 FE D0 06
0500 8A 09 40 4C 07 05 8A A6 FD A0 FF 60 A0 00 B1 FA
0510 4C 24 05 A5 FB 20 24 05 20 91 1F A5 FA 20 24 05
0520 20 91 1F 60 85 FC 4A 4A 4A 4A 20 35 05 A5 FC 20
0530 35 05 A5 FC 60 29 0F C9 0A 18 30 02 69 07 69 30
0540 4C 52 05 A9 0A 20 52 05 A9 0D 20 52 05 A2 FF 60
0550 A9 20 86 EC 20 8C 1E EA EA C9 0D D0 04 A9 08 D0
0560 0E C9 0A D0 04 A9 02 D0 06 29 3F AA BD C0 05 85
0570 EB 29 20 C5 EA F0 0E 85 EA A8 F0 04 A9 1B D0 02
0580 A9 1F 20 8D 05 A5 EB 20 8D 05 A6 EC 60 A2 06 8E
0590 42 17 20 B4 05 A0 04 4A 90 04 A2 07 D0 02 A2 06
05A0 8E 42 17 20 B4 05 88 10 EE A2 07 8E 42 17 A2 1D
05B0 20 B6 05 60 A2 13 8E 07 17 2C 07 17 10 FB 60 00
05C0 29 03 19 0E 09 01 0D 1A 14 06 0B 0F 12 1C 0C 18
05D0 16 17 0A 05 10 07 1E 13 1D 15 11 2D 00 3A 00 00
05E0 24 34 2C 29 2B 24 31 25 2F 32 39 31 2C 23 3C 3D
05F0 36 37 33 21 2A 30 35 27 26 38 2E 2B 24 3E 24 39
```

Bild 2. Hexadezimale Programmliste des Baudot-Monitors

Das Programm

Das Programm (*Bild 2*) ist nicht nur ein Monitor-Programm, welches in der Bedienung weitgehend identisch mit dem ASCII-Monitor im KIM-ROM ist. Es enthält auch für sämtliche Unterprogramme des KIM-ROMs, die mit dem ASCII-Fernschreiber zusammenarbeiten, jeweils ein entsprechendes Unterprogramm für Baudot-Fernschreiber, das in Anwenderprogrammen erscheinen kann. Innerhalb der Anwenderprogramme wird dabei ausschließlich mit ASCII-Zeichen gearbeitet, wie es bei Mikrocomputern üblich ist. In der *Tabelle* sind die Unterprogramme tabellarisch aufgeführt. Wenn ein Programm, welches für den Betrieb mit einem ASCII-Gerät geschrieben wurde, zusammen mit einem Baudot-Fernschreiber arbeiten soll, müssen die in der ersten Spalte der Tabelle im Programm vorkommenden Unterprogrammaufrufe durch jene in der 2. Spalte ersetzt werden.

Funktion des Monitorprogramms

Das Monitorprogramm ist in Page 4 und 5 untergebracht. Es wird an der Adresse 0400 gestartet und reagiert mit dem Ausdruck
KIM 0400 20.
Wenn die Adresse geändert werden soll, wird die neue Adresse eingetastet und dann die Leertaste gedrückt. Daraufhin werden vom Fernschreiber die neue Adresse und die darin stehenden Daten ausgedruckt. Sollen Daten geändert werden, so sind die neuen Daten und anschließend ein Punkt einzugeben. Das Monitorprogramm schreibt dann die neuen Daten in den Speicher und druckt die nächsthöhere Adresse aus. Wenn die nächsthöhere bzw. nächstniedrigere Adresse ausgedruckt werden soll, ist Wagenrücklauf bzw. Zeilenwechsel auszulösen. Wird „M" gedrückt, so folgt ein Sprung zum Systemprogramm des KIM-1. Wenn ein Programm an der zuletzt ausgedruckten Adresse gestartet werden soll, ist „G" zu betätigen. Bei der Eingabe von Adressen und Daten können Vornullen weggelassen werden.

Der Baudot-Monitor ist in der Bedienung mit dem im KIM vorhandenen ASCII-Fernschreib-Monitor fast identisch. Lochstreifenbefehle wurden in der Baudot-Version nicht vorgesehen, da Fernschreiber mit Lochstreifenzusatz ein Mehrfaches von dem kosten, was man für Modelle ohne Lochstreifenzusatz bezahlen muß, so daß die Speicherung von Programmen auf Kassettenrecordern preiswerter ist.

Das Programm arbeitet mit 50 Baud, so daß alle dem öffentlichen Telexnetz entstammende Fernschreiber benutzt werden können. Es kann auch in andere Bereiche des Speichers geladen werden, wenn die 3. Bytes von 3-Byte-Befehlen, falls diese 04 oder 05 lauten, entsprechend geändert werden. Dies läßt sich leicht mit einem Disassembler feststellen (vgl. FUNKSCHAU 1978, Heft 21).

Unterprogramme im Baudot-Monitor

ASCII-Startadresse	*Baudot-Startadresse*	*Name*	*Aufgabe*
1E2F	0543	CRLF	Wagenrücklauf, Zeilenwechsel
1E3B	0524	PRTBYT	Druckt Akkumulatorinhalt als 2 Hexadezimalzeichen aus
1E1E	0513	PRTPNT	Druckt Pointregister aus
1E9E	0550	OUTSP	Druckt einen Leerraum
–	050C	–	Druckt Daten unter der durch das Point-Register spezifizierten Adresse aus
1E5A	048A	GETCH	Holt ASCII-Zeichen vom Fernschreiber
1EA0	0552	OUTCH	Druckt Akkumulatorinhalt als ASCII-Zeichen
1F9D	0477	GETBYT	Holt 2 Hexadezimalzahlen vom Fernschreiber